WIND RIVER

Wind River® Trace for Wind River Workbench

HARDWARE REFERENCE

3.0

Corporate Headquarters
Wind River Systems, Inc.
500 Wind River Way
Alameda, CA 94501-1153
U.S.A.

toll free (U.S.): (800) 545-WIND
telephone: (510) 748-4100
facsimile: (510) 749-2010

For additional contact information, please visit the Wind River URL:

http://www.windriver.com

For information on how to contact Customer Support, please visit the following URL:

http://www.windriver.com/support

Wind River Trace for Wind River Workbench Hardware Reference, 3.0

6 Nov 07
Part #: DOC-16023-ND-00

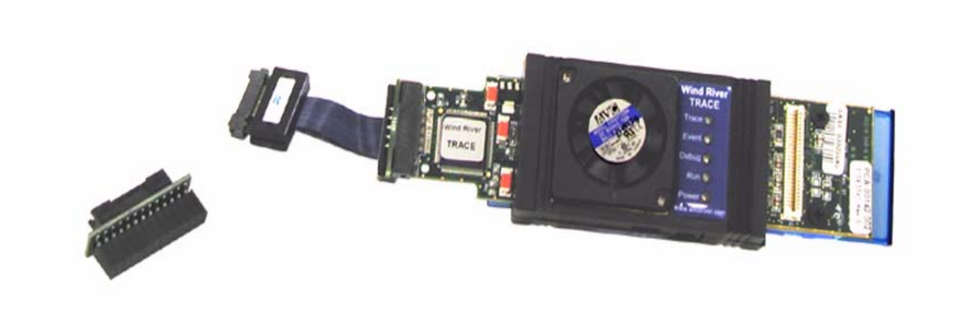
Contents

1
Overview

1.1 Introduction

This document describes the Wind River Trace tool. The Wind River Trace tool is used in conjunction with the Wind River ICE SX emulator, providing a real-time code execution trace.

Figure 1-1 **Wind River Trace**



The Wind River Trace captures a snapshot of your executing code to a memory array, at full speed. It saves up to hundreds of thousands of machine cycles, displaying addresses, instructions, and transferred data. The Wind River ICE SX emulator translates raw machine cycles to assembly code or C/C++ statements, and displays them in the **Trace** view in Wind River Workbench.

The Wind River Trace starts and stops trace collection based on user-defined triggering mechanisms. For example, you might instruct the Wind River Trace to capture every instance of an infrequent interrupt service routine.

Trace collection has no impact on the real time performance of your executing code.

This document outlines the information needed to use Wind River Trace, and includes the following chapters:

- *1. Overview* - Describes features of Wind River Trace and provides necessary safety information and contact information.
- *2. Hardware Setup*- Describes physical connection information and explains connecting power.
- *3. Using Wind River Trace*- Describes how to use the **Trace** view in Wind River Workbench to trace code.

This document also includes the following appendices:

- *A. 38-pin Mictor Connector (AMCC 405 and 440 Processors)*- Pinout information for 405 and 440 processors.
- *B. 26-pin BDM Connector (ColdFire Processors)*- Pinout information for ColdFire processors.

Other Resources

For information on Wind River Workbench, see the *Wind River Workbench User's Guide*.

For information on Workbench low-level commands, see the *Wind River Workbench for On-Chip Debugging Command Reference*.

For information on configuring Wind River emulators, see the *Wind River Workbench for On-Chip Debugging Configuration Options Reference*.

For information on on-chip debugging, see the *Wind River Workbench for On-Chip Debugging User Tutorials*.

For information on the Wind River ICE SX, see the *Wind River ICE SX for Wind River Workbench Hardware Reference.*

For Tech Tips and Application Notes dealing with Workbench, the Wind River ICE SX and Wind River Probe tools, and various processor-specific issues, go to **http://www.windriver.com/support**.

1.2 Wind River Trace Features

Wind River Trace adds a high-performance trace buffer to Wind River ICE SX, without compromising the functionality of the hardware debug tools. Wind River Trace captures the trace output emitted by the processor via the processor's real-time trace interface.

Wind River Trace operates with the Wind River Workbench debugger, and is designed primarily for software development. It offers real-time trace with a high-depth trace buffer, which is useful in the integration phase of projects, as well as in the field or for remote monitoring.

1.2.1 AMCC 405 and 440 Processors (Formerly IBM 405 and 440)

Wind River Trace includes the following features:

- A non-intrusive interface for tracing code execution in real-time, even when code is running out of the internal instruction cache

- Real-time trace with the AMCC 405 or 440 trace clock running at up to 200 MHz (since the trace clock runs at half the core clock frequency for 405, and one quarter the core clock frequency for 440, the AMCC 405 core clock can run at up to 400 MHz, and the AMCC 440 core clock can run at up to 800 MHz.) The capture frequency may increase with future development. Contact **http://www.windriver.com/support** for information on all updates.
- A 512KB trace memory and 512KB of Time Stamp memory that can capture up to 640,000 lines of code (depending on the state of the processor's instruction cache and data cache)
- Filtered trace for tracing selected areas of code

1.2.2 Freescale ColdFire Processors (Formerly Motorola ColdFire)

- A non-intrusive interface for tracing code execution in real-time, even when code is running out of the internal instruction cache.
- A small debug exception routine is required for filtered trace operations; impact on code execution is minimal. The debug exception routine is loaded in a predefined workspace area of RAM.
- Can capture real-time trace with the ColdFire trace clock running at up to 200 MHz (since the trace clock runs at the core clock frequency for the Version 2 and Version 3 core processors, and at half the core clock frequency for the Version 4 core processors, the Version 2 and Version 3 core clocks can run at up to 200 MHz, and the Version 4 core clock can run at up to 400 MHz.)
- Contains a 512KB trace memory and 512KB of Time Stamp memory that can capture up to 380,000 lines of code (depending on the processor type and the state of the processor's instruction, data, and branch caches.)
- Supports filtered trace for tracing selected areas of code.

1.3 Wind River ICE Features

The Wind River ICE SX is used for high-performance source-level debugging and early hardware testing. The Wind River ICE SX has both serial and 10/100 BaseT Ethernet capability from host to target.

The Wind River ICE SX utilizes On-Chip Debugging (OCD) services embedded in the target microprocessor (BDM/JTAG/EJTAG). These OCD services are a set of debugging services that reside in the chip's micro-code. When accessed, they provide complete control of the target microprocessor.

There is no hardware or software intrusion into the system, and these debug services do not require any target interrupts, RAM, ROM, or RS-232 communications. All interaction between the Wind River ICE SX and the processor runs exclusively through the OCD link. The result is that the tools are effective for the entire development process, even before board-level peripherals are stable.

1.4 Safety Information

There are some basic safety precautions that should be observed when using your Wind River Trace. Following these precautions will help you to avoid injury and prevent damage to the tools and/or any products connected to them. To avoid any hazardous conditions, use this product only as specified. For the purposes of this manual, *Warning* and *Caution* symbols are used to mean the following:

WARNING: Warning statements indicate conditions that could result in injury or loss of life and describe how to avoid them.

CAUTION: Caution statements indicate conditions that could result in damage to this product or other property and describe how to avoid them.

Precautions to avoid Injury

WARNING: Do not operate in wet or damp environments or outside of the recommended operating conditions. This product is intended for indoor use only.

Only use the power cord specified for this product with a properly grounded power outlet.

Do not operate this product in an explosive atmosphere.

Do not operate the product if damaged. Have a qualified service person inspect damaged equipment before use.

Precautions to Avoid Property Damage

⚠ **CAUTION:** Take precautions against electrostatic discharge as it may damage some components.

Use care in handling, as delicate components can be easily damaged.

Provide proper ventilation to prevent the product from overheating.

2 Hardware Setup

2.1 Introduction

This chapter includes information related to hardware connections with Wind River Trace.

2.2 Connection Schemes

Different methods are available for making the physical connections between the target and the Wind River Trace/Wind River ICE SX tool, depending on processor type. The preferred connection scheme is specified when purchased. The following two sections describe the connection schemes in detail.

2.2.1 AMCC 405 and 440 Processors

The connection to the AMCC 405 or 440 target board is through a 38-pin Mictor connector, which contains all the run control and trace functionality in a single connector. A short coaxial ribbon cable, with Mictor connectors, is provided to eliminate possible interference problems with connecting to the target board. The trace signals are high frequency, so when routing the signals, make sure to keep them as short as possible.

For pin-out information for the 38-pin Mictor connector, and recommended manufacturer's part numbers for the connector, see *A. 38-pin Mictor Connector (AMCC 405 and 440 Processors)*.

2.2.2 ColdFire Processors

The connection to the ColdFire target board is through a 26-pin BDM connector. A short coaxial ribbon cable, with Mictor connectors, is provided to eliminate possible interference problems with connecting to the target board. For the pin-out information for the 26-pin BDM connector, and the recommended manufacturer's part number for the connectors, see *B. 26-pin BDM Connector (ColdFire Processors)*.

2.3 Trace Signal Requirements

2.3.1 Signal requirements for the AMCC 405 processor

Wind River Trace for the AMCC 405 requires that all of the trace signals from the 405 processor be properly connected to the trace interface 38-pin Mictor connector. The trace signals required are TRCCLK, TS1E, TS1O, TS2E, TS2O, TS3, TS4, TS5, TS6, and GND. The trace signals are high-speed signals, so be extremely careful when connecting these signals from the AMCC 405 processor to the trace interface Mictor connector. The AMCC 405 trace pins are dual-mode multiplexed pins that can be configured as trace signals or as GPIO[0:9] signals. The Wind River ICE SX will automatically configure these pins to be trace signal pins when the configuration option **Acquire Trace on GO (TRCAQU)** is set to **ON**. The AMCC 405 trace signals are enabled only when the AMCC 405 is actually executing code.

NOTE: These pins must not be re-configured as GPIO pins by the application code.

2.3.2 Signal requirements for the AMCC 440 Processor

Wind River Trace for the AMCC 440 requires that all of the trace signals from the 440 processor be properly connected to the trace interface 38-pin Mictor connector. The trace signals required are TRCCLK, TS0, TS1, TS2, TS3, TS4, TS5, TS6, ES0, ES1, ES2, ES3, ES4, BS0, BS1, BS2, and GND. The trace signals are high-speed signals, so be extremely careful when connecting these signals from the AMCC 440 processor to the trace interface Mictor connector. The AMCC 440 trace pins are dual-mode multiplexed pins that can be configured as trace signals or as GPIO[18:31] signals. Wind River ICE SX automatically configures these pins to be trace signal pins when the configuration option **Acquire Trace on GO (TRCAQU)** is set to **ON**. The AMCC 440 trace signals are enabled only when the AMCC 440 is actually executing code.

NOTE: These pins must not be re-configured as GPIO pins by the application code.

2.3.3 Signal Requirements for ColdFire Processors

Wind River Trace requires that all trace signals from the ColdFire processor be properly connected to the trace interface 26-pin BDM connector.The trace signals required are described in Table 2-1.

Table 2-1 **ColdFire Trace Signal Requirements**

Processor	Trace Signals
MCF5202, MCF5204, MCF5206, MCF5206E	PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, DDATA3, CLK_CPU, VCC_CPU, GND
MCF5249, MCF5249L, MCF5250, MCF5251, MCF5272, MCF5307, MCF5307A, MCF5307B, MCF5327, MCF5328, MCF5329, MCF5372, MCF5372L, MCF5373, MCF5373L	PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, DDATA3, PSTCLK, VCC_CPU, GND

Table 2-1 **ColdFire Trace Signal Requirements**

Processor	Trace Signals
MCF5207, MCF5208, MCF5211, MCF5212, MCF5213, MCF5214, MCF5216, MCF5232, MCF5233, MCF5234, MCF5235, MCF5270, MCF5271, MCF5274, MCF5274L, MCF5275, MCF5275L, MCF5280, MCF5281, MCF5282	PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, DDATA3, CLKOUT, VCC_CPU, GND
MCF52221, MCF52223, MCF52230, MCF52231, MCF52233, MCF52234, MCF52235	PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, DDATA3, PSTCLK, VCC_CPU, GND
MCF5407, MCF5470, MCF5471, MCF5472, MCF5473, MCF5474, MCF5475, MCF5480, MCF5481, MCF5482, MCF5483, MCF5484, MCF5485	PSTDDATA0, PSTDDATA1, PSTDDATA2, PSTDDATA3, PSTDDATA4, PSTDDATA5, PSTDDATA6, PSTDDATA7, PSTCLK, VCC_CPU, GND

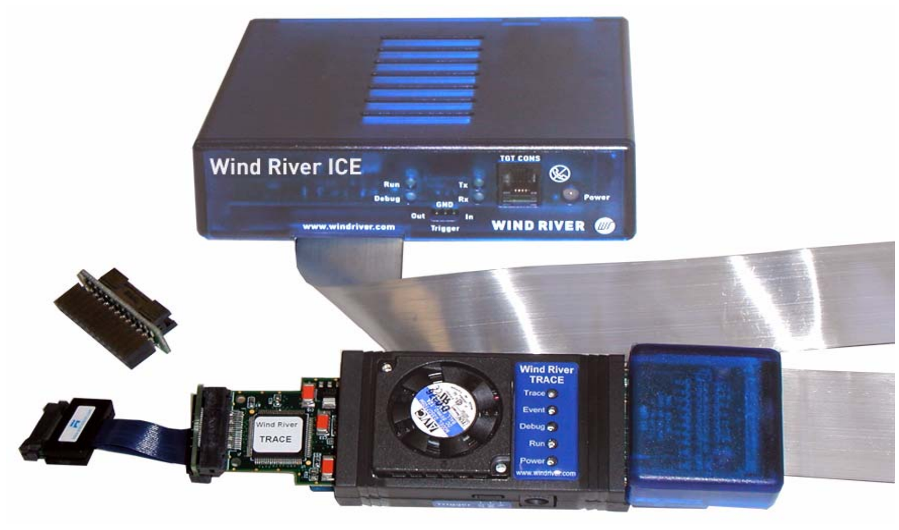
The trace signals are high-speed signals, so be extremely careful when connecting these signals from the ColdFire processor to the trace interface BDM connector. Some ColdFire trace signals are multiplexed pins that can be configured as trace signals or as GPIO signals.

NOTE: These trace signals must not be re-configured as GPIO pins by the application code.

2.4 Wind River Trace with Wind River ICE SX

The Wind River Trace unit is shipped already connected to the Wind River ICE SX unit, as shown in Figure 2-1.

Figure 2-1 **Wind River Trace with Wind River ICE SX**



Wind River ICE SX connects to Wind River Trace via a 51-pin extension cable.

If you order the Wind River Trace as a separate product, it will have to be attached to your Wind River ICE SX before use.

⚠ **CAUTION:** Wind River recommends that you return your Wind River ICE SX to a Wind River service representative to have the Wind River Trace unit attached to it.

To connect the Wind River Trace to the Wind River ICE SX, use the following steps:

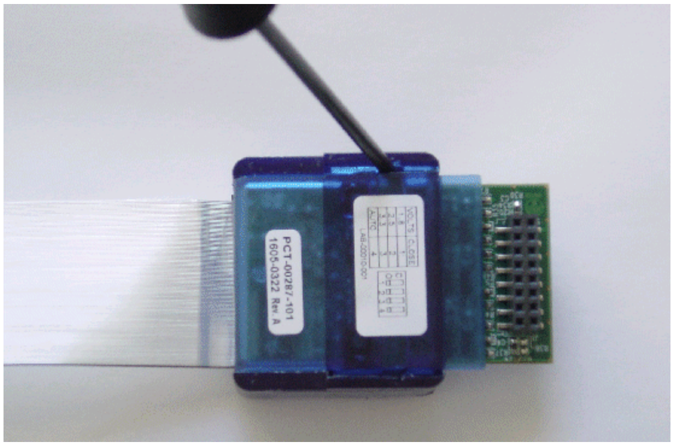
1. Remove the personality module from the Wind River ICE SX.

   ⚠ **CAUTION:** Before attempting to remove the personality module on the ICE unit, make sure that the power on the ICE unit is **OFF**, and that it is disconnected from its power supply.

   On the underside of the target end of the 51-pin cable, you will see a small rectangular cover that clips the personality module into place.

2. Pry the module out of its position by gently pushing a small screwdriver into the open grooves located on the sides of the cover, as shown in Figure 2-2.

Figure 2-2 **Removing the Personality Module**

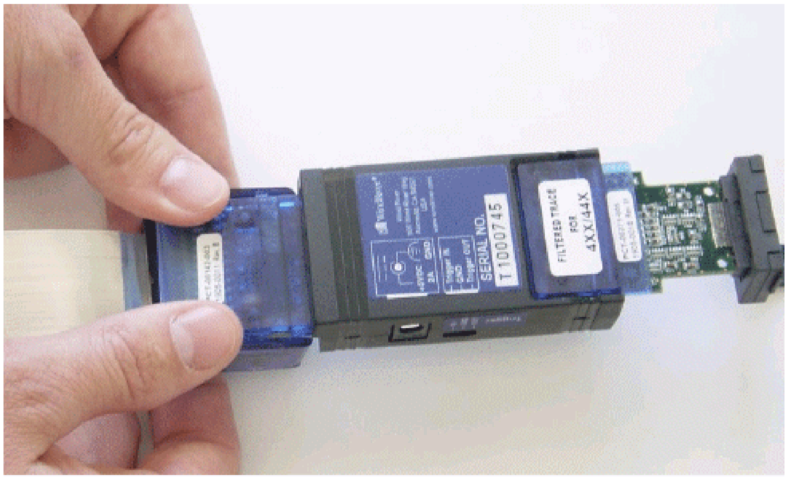


3. To attach the Wind River Trace, align the rear edge of the connector with the rear edge of the target end of the 51-pin cable.

   The fan on the Wind River Trace should be facing away from you.

   When the rear edges are aligned, as shown in Figure 2-3, the grooves on both the connector and the 51-pin cable are also aligned.

Figure 2-3 **Attaching the Trace Unit**



4. Press both thumbs on each side of the connector, and snap the Wind River Trace into place.

   When inserting, press down on both sides of the connector with equal force.

   When the Trace unit is properly inserted, it snaps into place and is solidly attached to the cable.

A 4xx/44x or ColdFire filtered trace cable is shipped already attached to the Wind River Trace unit.

2.4.1 Applying Power to Wind River Trace and Wind River ICE SX

The Wind River Trace unit includes a dedicated 5V power supply that should be mated with a power cord appropriate for the area. For Wind River specified part numbers, see Table 2-2. The Wind River ICE SX unit includes a standard, 5-pin circular DIN connector power supply that converts standard 110-240 VAC 50/60 Hz to 5V DC at 5A. The DC input connection is located on the back of the Wind River ICE SX unit. For more information, see Table 2-2.

WARNING: To avoid electrical shock and possible loss of life, do not disassemble the power supply. The power supply contains no user serviceable parts and should not be disassembled.

Figure 2-4 **Wind River ICE SX and Wind River Trace Power Inputs**

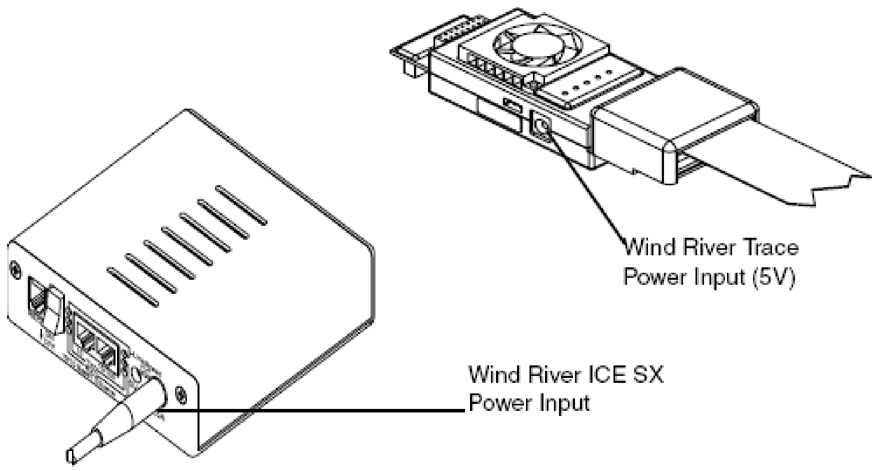


Table 2-2 **Power Supply and Cord Part Numbers**

Wind River Part Number	**Description**
PWR-R00009	Power supply for Wind River Trace (5V)
PWR-R00010-001	Power supply for Wind River ICE SX
CAB-R00003	Power Cord, USA
CAB-R00004	Power Cord, Europe
CAB-R00005	Power Cord, UK
CAB-R00018	Power Cord, Israel
CAB-R00019	Power Cord, Australia

NOTE: The appropriate power supply should be matched with the cable from the above list that applies to your location.

3
Using Wind River Trace

3.1 Getting Started

This chapter explains how to configure Wind River Workbench to read trace information from the Wind River Trace unit. For information on Workbench, please refer to the *Wind River Workbench User's Guide*.

NOTE: If you have not yet installed Wind River Workbench, you should install it before continuing.

Before you can use the Wind River Trace, you must set a new environment variable called **DFW_OPCODE_READER** on your host system to 1. To set this variable, use the following procedure:

Windows Hosts:

You must have administrator privileges to set environment variables.

1. Select **Start > Control Panel > System**.
2. In the dialog that appears, click on the **Advanced** tab.
3. Click **Environment Variables** to open the **Environment Variables** dialog.
4. Click **New**.

   The **New Variable** dialog appears.
5. In the **Variable Name:** field, enter **DFW_OPCODE_READER**.
6. In the **Variable Value:** field, enter 1.
7. Click **OK**.
8. Exit the Control Panel.

Linux/Solaris Hosts:

1. Login as root.
2. Set the variable using the following commands:

   If you are using the C Shell, enter

   ```
   setenv DFW_OPCODE_READER 1
   ```

   If you are using **bash**, enter

   ```
   export DFW_OPCODE_READER=1
   ```
3. Exit root.

3.2 Establishing Communications

The Wind River Trace operates only in conjunction with a Wind River ICE SX. To establish communications with your Wind River ICE SX, use the following steps.

First, open Workbench according to the method for your host computer.

Linux/Solaris Hosts

From your installation directory, issue the command

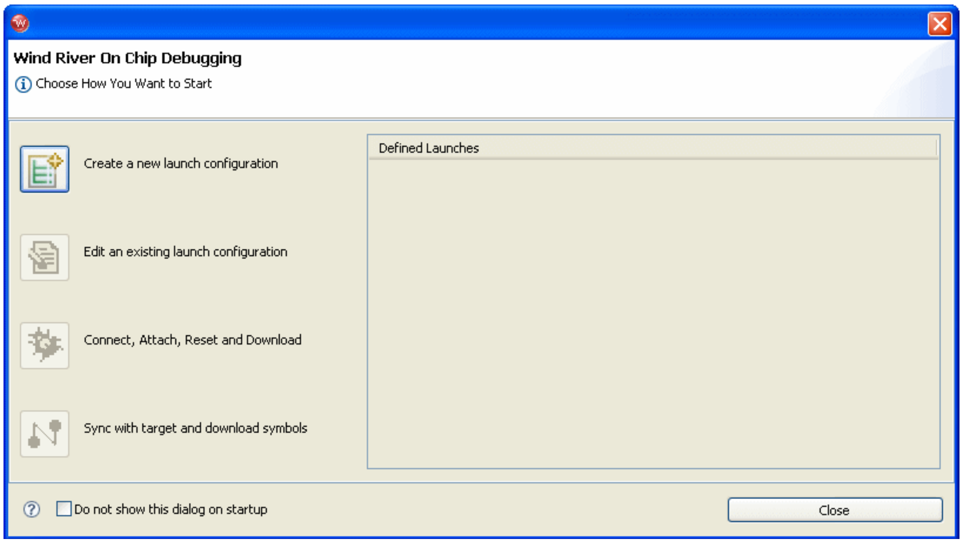
```
$ ./startWorkbench.sh
```

Windows Hosts

Select **Start > All Programs > Wind River > Wind River Workbench** *version*.

On Windows hosts, Workbench prompts you to specify a workspace location. Linux hosts use the default location *installDir*/**workspace**.

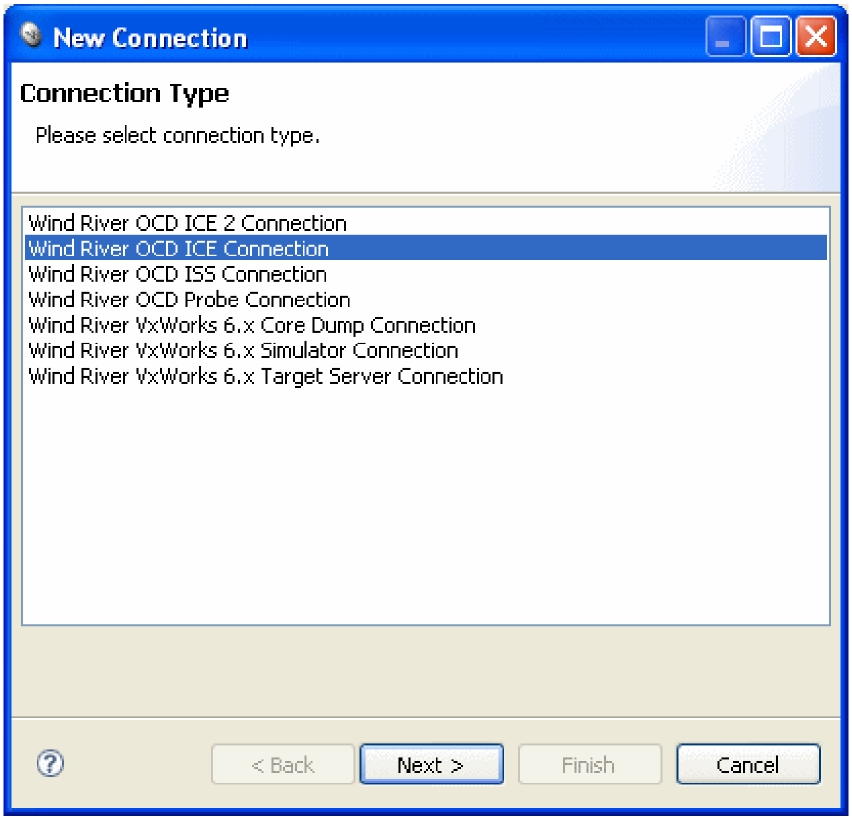
When Workbench opens, the **Quick Target Launch** dialog appears.


Wind River On Chip Debugging
Choose How You Want to Start

Create a new launch configuration
Edit an existing launch configuration
Connect, Attach, Reset and Download
Sync with target and download symbols
Defined Launches
Do not show this dialog on startup
Close

1. Select **Create a new launch configuration**.

   The **Connection Type** dialog appears.


New Connection

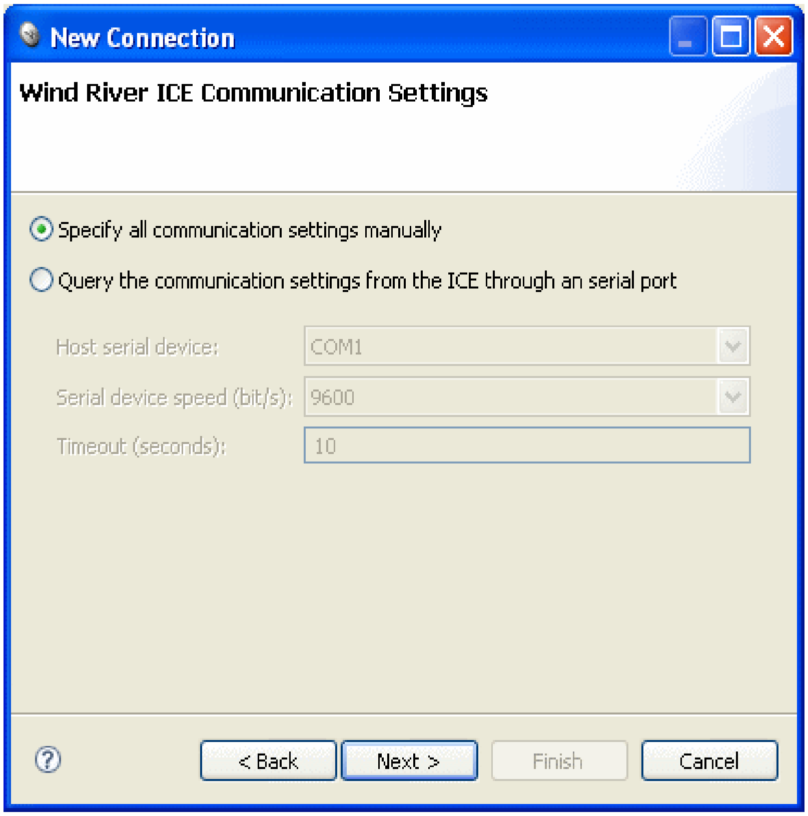
Connection Type

Please select connection type.

Wind River OCD ICE 2 Connection
Wind River OCD ICE Connection
Wind River OCD ISS Connection
Wind River OCD Probe Connection
Wind River VxWorks 6.x Core Dump Connection
Wind River VxWorks 6.x Simulator Connection
Wind River VxWorks 6.x Target Server Connection

2. Choose **Wind River OCD ICE Connection** from the list of options and click **Next**.

The **Communication Settings** dialog appears.



(● Specify all communication settings manually)
○ Query the communication settings from the ICE through an serial port

Host serial device: COM1
Serial device speed (bit/s): 9600
Timeout (seconds): 10

To configure communication settings manually, **see** *Configuring Communication Settings Manually*, p.19**.** To configure communication settings through a serial port, see *Configuring Communication Settings Manually*, p.19.

Configuring Communication Settings Manually

NOTE: If you choose this option you will need to know either the network name of the emulator or its IP address. For information on assigning these values, see the *Wind River ICE SX for Wind River Workbench Hardware Reference*.

3. Check the **Specify all communication settings manually** box and click **Next.**

The **Emulator Settings** dialog appears.

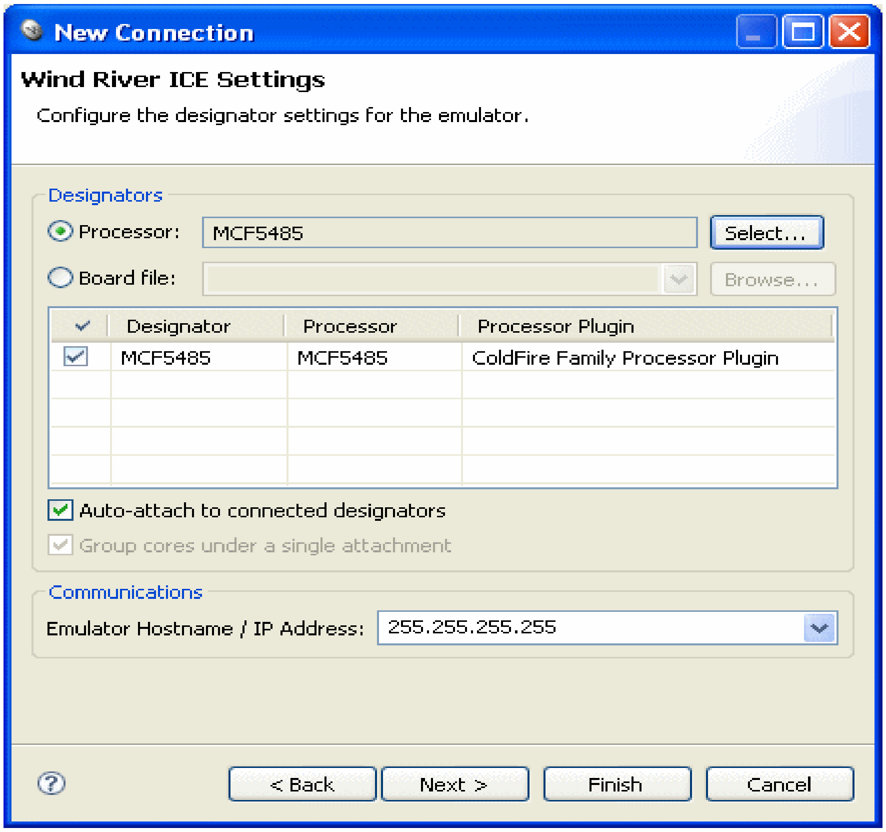
4. In the **Designators** area, enter your target processor type in the **Processor** field, or click **Select** to choose from a list of available processor types.

   If you are using multiple processors, or if you have other devices besides your processor on your JTAG scan chain, you must specify a board file in the **Board File** field. Check the **Board File** radio button and click **Browse** to navigate to your board file.

   Either choice will populate the field below the **Board File** field with a summary description of your board.

5. In the **Communications** area, fill in the **IP Address** field with the IP address you have assigned to your ICE unit.

   This example shows the **Emulator Settings** dialog with values entered for a ColdFire MCF5485 target processor.


New Connection

Wind River ICE Settings

Configure the designator settings for the emulator.

Designators

(● Processor: MCF5485) [Select...]

( ) Board file: [Browse...]

✓	Designator	Processor	Processor Plugin
✓	MCF5485	MCF5485	ColdFire Family Processor Plugin

[✓] Auto-attach to connected designators

[✓] Group cores under a single attachment

Communications

Emulator Hostname / IP Address: 255.255.255.255

6. When you have entered the correct processor or board file and IP address, click **Next**.

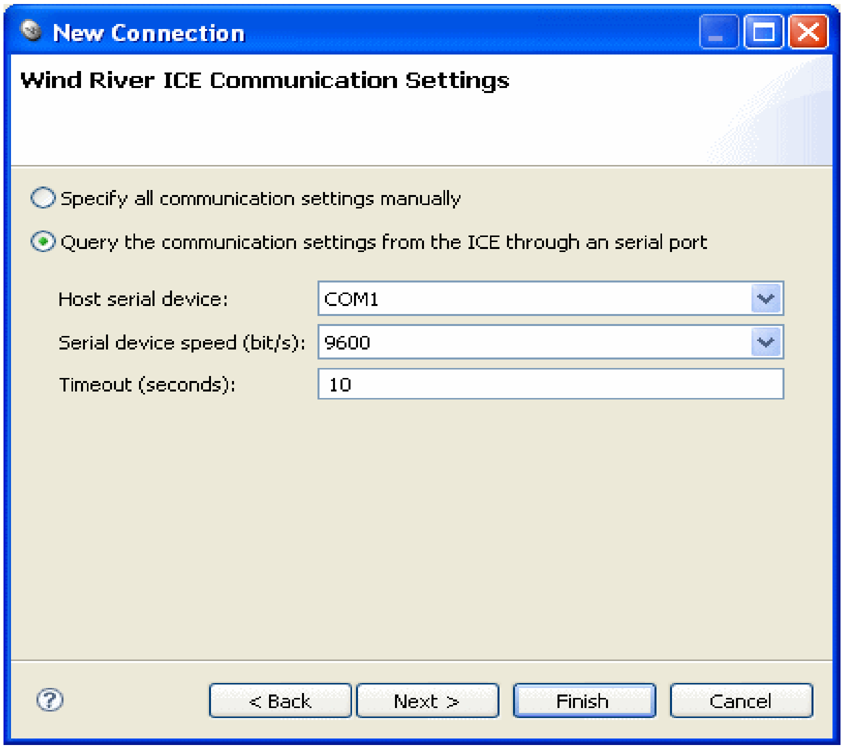
   The **Target Operating System Settings** dialog appears. Proceed to Step 7.

Configuring Communication Settings Through a Serial Port

If you choose to make your connection using the serial port, make sure that a serial cable is connected between Wind River ICE SX and your host computer.

NOTE: A direct serial cable is required to create your connection this way. If you do not have a direct serial connection between your host and emulator, you must configure your ICE settings manually.

a. Select **Query the communication settings from the ICE through a serial port** and click **Next.** The serial settings fields become active.

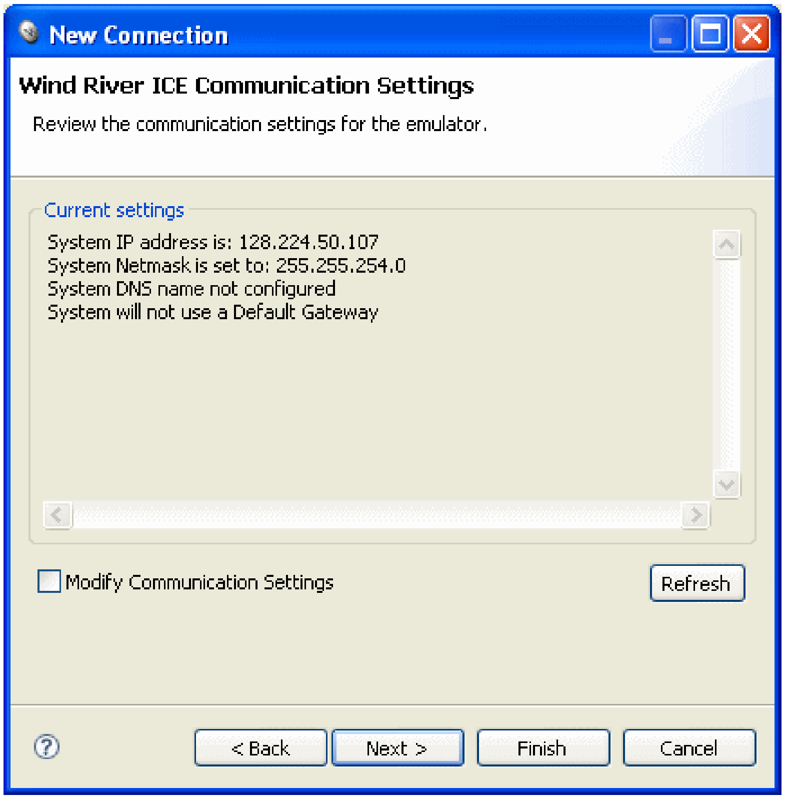


Wind River ICE Communication Settings

( ) Specify all communication settings manually
(●) Query the communication settings from the ICE through an serial port

Host serial device: COM1
Serial device speed (bit/s): 9600
Timeout (seconds): 10

b. Use these fields to select the serial port you want to connect to, and set a timeout value in seconds. Communication settings, such as the emulator's dynamically assigned IP address and other settings, are retrieved and displayed automatically.



Wind River ICE Communication Settings

Review the communication settings for the emulator.

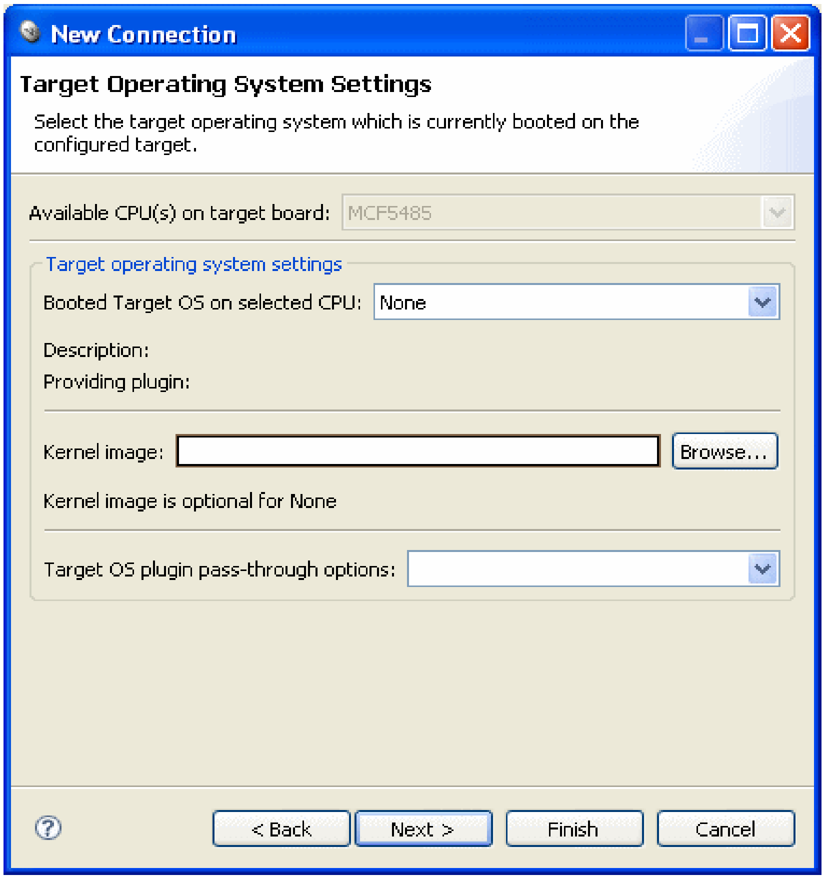
Current settings

```
System IP address is: 128.224.50.107
System Netmask is set to: 255.255.254.0
System DNS name not configured
System will not use a Default Gateway
```

The retrieved settings can also be modified (select the **Modify Communication Settings** checkbox); that is, you can reconfigure the emulator's communication settings using a serial connection.

c. Click **Next**.

The **Target Operating System Settings** dialog appears. Proceed to Step 7.


New Connection

Target Operating System Settings

Select the target operating system which is currently booted on the configured target.

Available CPU(s) on target board: MCF5485

Target operating system settings

Booted Target OS on selected CPU: None

Description:

Providing plugin:

Kernel image:

Kernel image is optional for None

Target OS plugin pass-through options:

7. In the **Booted Target OS on selected CPU** field, select the operating system that is running on your target processor. The default is **None**.

8. Next to the **Kernel Image** field, click **Browse** to navigate to the kernel image you wish to specify. If you selected **None** in the previous step, you do not need to specify a kernel image.

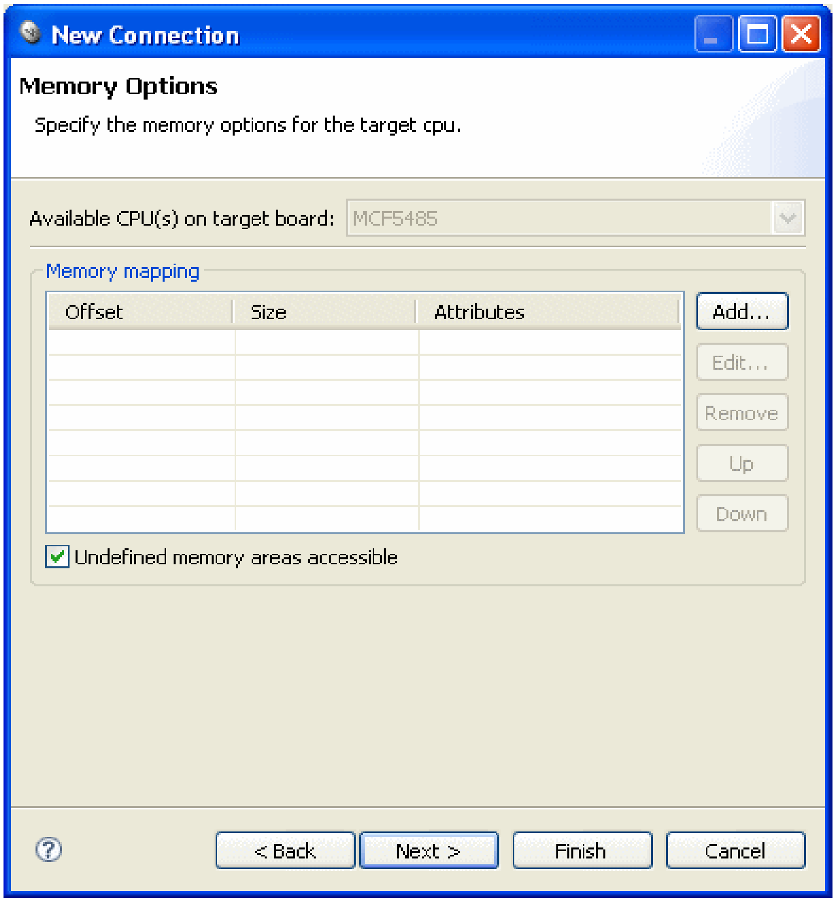
9. If you are using a Linux plug-in specify the pass-through options in the **Target OS Pass-Through Options** field. If you are not using a Linux plug-in, skip this step.

   Options are passed as pairs in the format *name=value*. Separate options with a comma. The following options are available:

- **notasklist=1**: Never fetch process list.
- **noautomodules=1**: Do not plant internal breakpoints to do automatic kernel module load/unload detection. When this option is specified, you must manually refresh to see an updated module list.
- **noloadcheck=1**: Do not issue gophers until the hardware breakpoint is used to detect kernel load triggers. This option is for "sensitive" boards that don't accept access until the kernel loads and sets up memory mapping.
- **loaddetectloc=***symbol* **or** *address*: Set the hardware breakpoint used to detect kernel load at *symbol* (for example, **loaddetectloc=start_kernel**) or *address* (for example, **loaddetectloc=0x1000**). If you do not specify a symbol or address, Workbench uses a default. For most architectures the default is **start_kernel**; for PowerPC targets, the default is **0x0**.

10. Click **Next**.

    The **Memory Options** dialog appears.

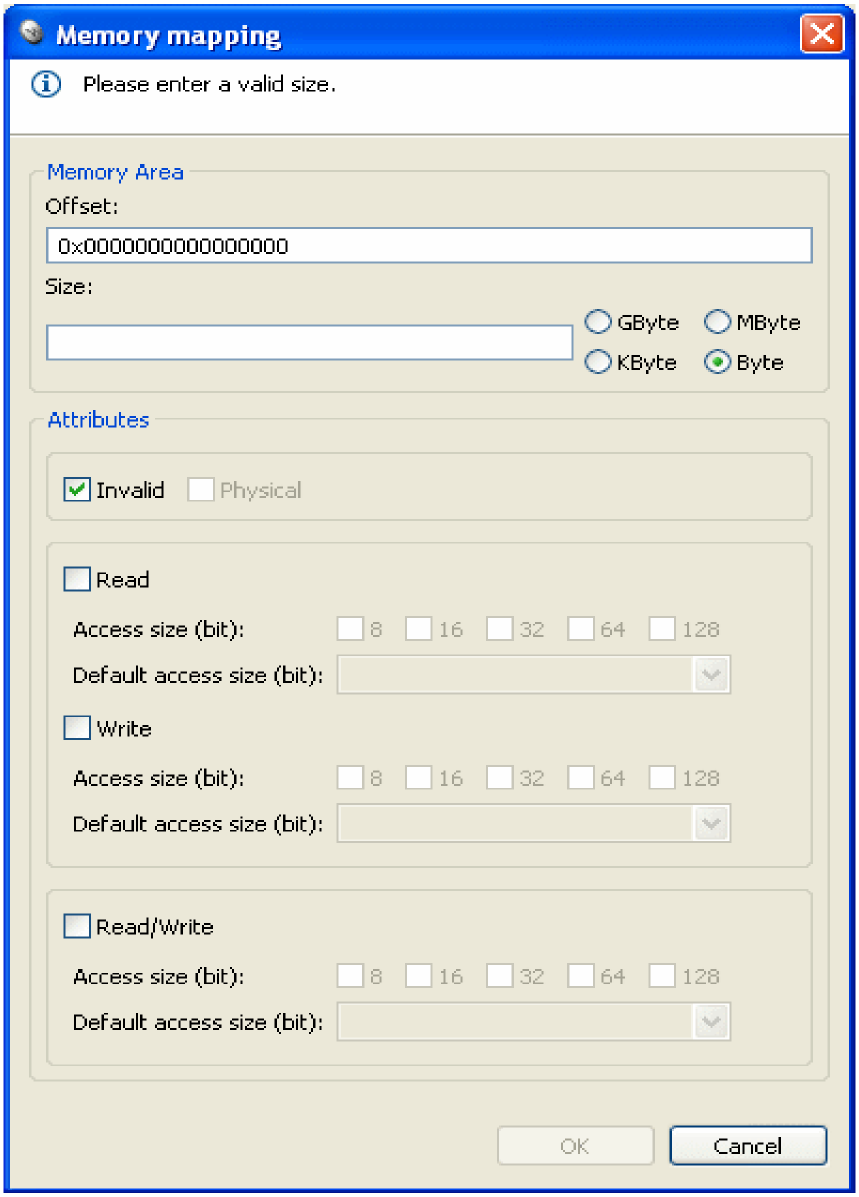

New Connection
Memory Options
Specify the memory options for the target cpu.
Available CPU(s) on target board: MCF5485
Memory mapping
Offset
Size
Attributes
Add...
Edit...
Remove
Up
Down
Undefined memory areas accessible
< Back
Next >
Finish
Cancel

Use the **Memory Options** dialog to specify how memory on the target is partitioned, and what the attributes of the particular memory regions are.

NOTE: The **Memory Options** dialog is only necessary for Linux or other non-VxWorks target operating systems.

To specify an area of memory, click **Add**.

The **Set Memory Map** dialog appears.


Memory mapping

Please enter a valid size.

Memory Area

Offset: 0x0000000000000000

Size: [ ] GByte / MByte / KByte / [✓ Byte]

Attributes

[✓] Invalid [ ] Physical

[ ] Read
Access size (bit): [ ] 8 [ ] 16 [ ] 32 [ ] 64 [ ] 128
Default access size (bit):

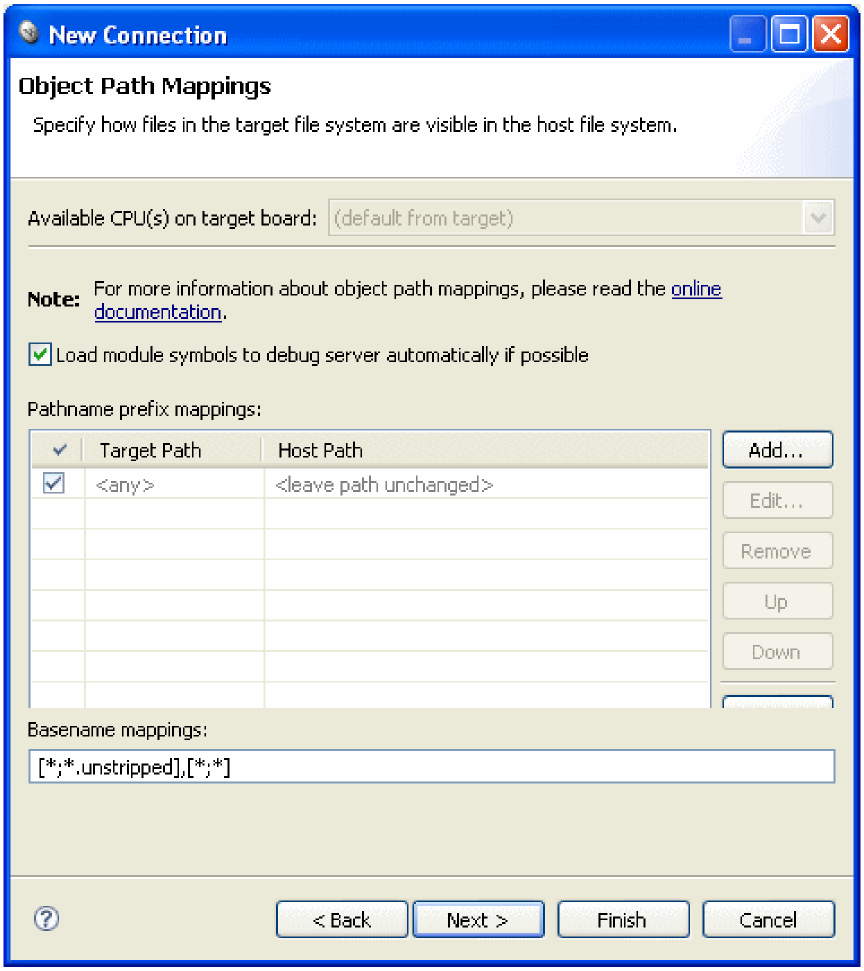
[ ] Write
Access size (bit): [ ] 8 [ ] 16 [ ] 32 [ ] 64 [ ] 128
Default access size (bit):

[ ] Read/Write
Access size (bit): [ ] 8 [ ] 16 [ ] 32 [ ] 64 [ ] 128
Default access size (bit):

Use the **Set Memory Map** dialog to specify which memory areas are read-only, read-write, or write-only, and to specify the access width Workbench should use to read the data from those regions.

11. Click **Next**.

    The **Object Path Mappings** dialog appears.


New Connection
Object Path Mappings
Specify how files in the target file system are visible in the host file system.

Available CPU(s) on target board: (default from target)

Note: For more information about object path mappings, please read the online documentation.

✓ Load module symbols to debug server automatically if possible

Pathname prefix mappings:

✓	Target Path	Host Path
✓	<any>	<leave path unchanged>

Basename mappings: [*;*.unstripped],[*;*]

Use the **Object Path Mappings** dialog to specify how files in the target file system are visible in the host file system.

Workbench uses object path mappings in two ways:

- They allow the debugger to find symbol files for processes created on the target by creating a correspondence between a path on the target and the appropriate path on the host.
- Workbench also uses object path mappings to calculate target paths for processes that you want to launch by browsing to them with a host file system browser.

By default, the debug server attempts to load all of a module's symbols each time a module is loaded. In the rare cases where you want to download a module or start a process without loading the symbol file, clear the **Load module symbols to debug server automatically if possible** checkbox.

The **Pathname prefix mappings** field maps target path prefixes to host paths. Always use full paths, not relative paths.

In most cases Workbench provides correct defaults. If necessary, click **Add** to add new mappings, or select existing mappings and click **Edit** to modify them.

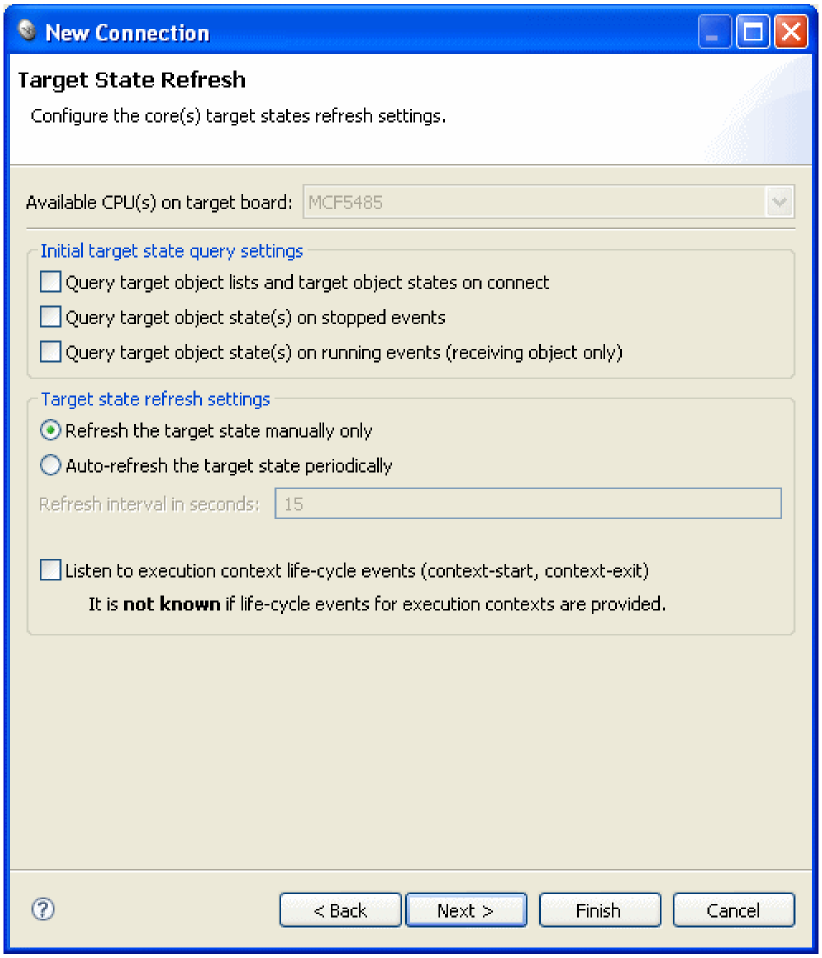
NOTE: You cannot edit the supplied default mappings.

Use square brackets to enclose each mapping of target file basenames (left element) to host file basenames (right element), separated by a semi-colon (**;**). Mapping pairs (in square brackets) are separated by commas. You can use an asterisk (*) as a wildcard.

For example, if debug versions of files are identified by the extension ***.unstripped**, then the mapping **[*;*.unstripped]** will ensure that the debugger loads *yourApp***.unstripped** when *yourApp* is launched on the target.

12. Click **Next**.

The **Target State Refresh** dialog appears.



Target State Refresh

Configure the core(s) target states refresh settings.

Available CPU(s) on target board: MCF5485

Initial target state query settings

- [ ] Query target object lists and target object states on connect
- [ ] Query target object state(s) on stopped events
- [ ] Query target object state(s) on running events (receiving object only)

Target state refresh settings

- [✓ Refresh the target state manually only]
- Auto-refresh the target state periodically

Refresh interval in seconds: 15

- [ ] Listen to execution context life-cycle events (context-start, context-exit)

It is **not known** if life-cycle events for execution contexts are provided.

Since retrieving status information from the target leads to considerable target traffic, this page allows you to configure how often and under what conditions the information displayed in the **Remote Systems** view is refreshed.

You can change these settings later by right-clicking the target connection and selecting **Refresh Properties**.

In the **Available CPU(s) on target board** field, Workbench identifies the target CPU. In rare cases, a close variant might be misidentified, so you can manually set the CPU here.

Use the **Initial target state query settings** checkboxes to specify whether Workbench should query the target on connect, on stopped events, and/or on running events. You can select any or all of these options.

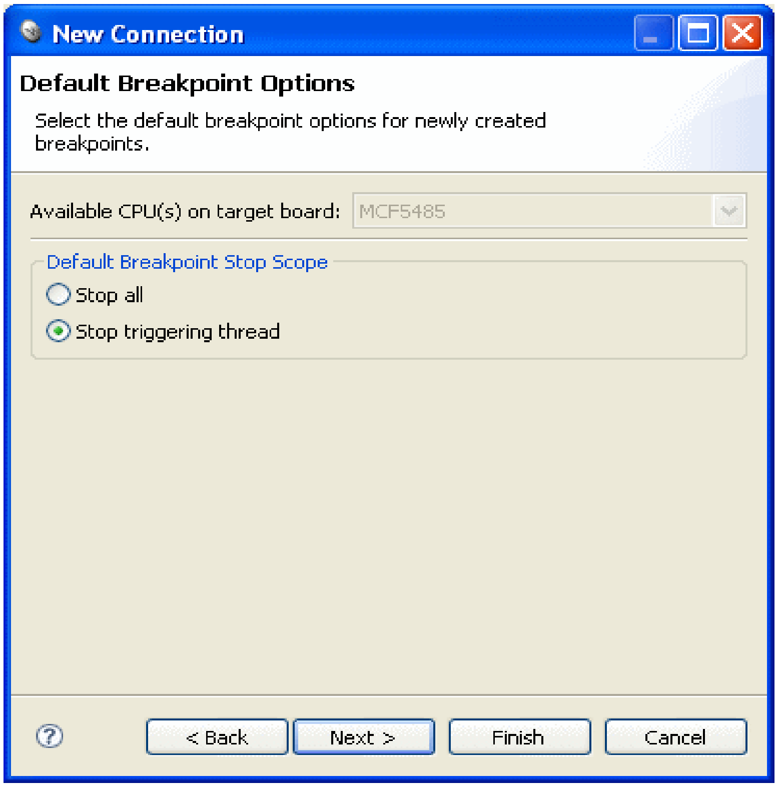
Use the **Target state refresh settings** to specify whether Workbench should auto-refresh the target state, or if it should only refresh when you refresh manually. You can also specify the auto-refresh interval.

If you want newly created execution contexts (such as tasks or processes) to be automatically added or removed from the **Remote Systems** view tree, select the **Listen for execution context life-cycle events** checkbox.

Your target may not provide information on life-cycle events for execution contexts. If it does not, selecting this checkbox has no effect. However, the Workbench backend has no way of detecting whether your target provides life-cycle events or not, so Workbench does not warn you that they are not provided. The only way to tell whether these events are provided is to select the checkbox and look for the events in the **Remote Systems** view tree.

13. Click **Next**.

    The **Default Breakpoint Options** dialog appears.



Default Breakpoint Options

Select the default breakpoint options for newly created breakpoints.

Available CPU(s) on target board: MCF5485

Default Breakpoint Stop Scope

( ) Stop all

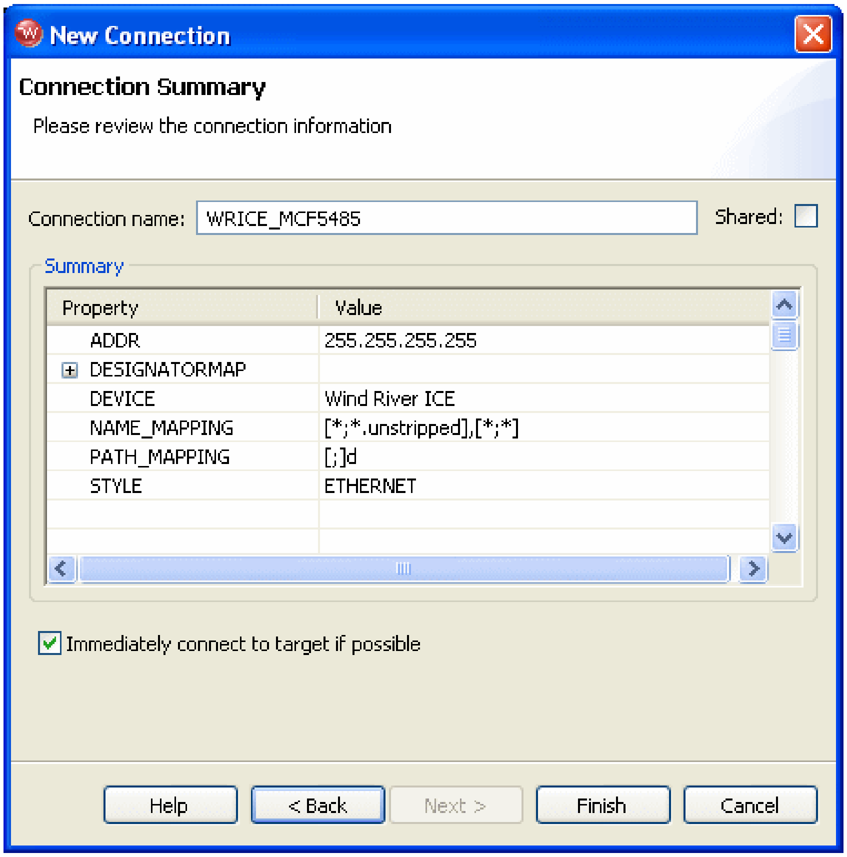
(●) Stop triggering thread

Use this dialog to set default breakpoint options for newly created breakpoints.

- Select **Stop all** if you want the breakpoint to stop all threads.
- Select **Stop triggering thread** if you want the breakpoint to stop only the thread that triggered it.

14. Click **Next**.

The **Connection Summary** dialog appears. Inspect the displayed values to make sure they are correct.



Connection Summary

Please review the connection information

Connection name: WRICE_MCF5485

Shared: ☐

Summary

Property	Value
ADDR	255.255.255.255
⊞ DESIGNATORMAP	
DEVICE	Wind River ICE
NAME_MAPPING	[*;*.unstripped],[*;*]
PATH_MAPPING	[;]d
STYLE	ETHERNET

☑ Immediately connect to target if possible

15. Check the displayed values to make sure they are correct.

    To connect to your target now, select **Immediately connect to target if possible**.

16. If you want to share your target connection, select **Shared**.

    This option serves a dual purpose:

- When you define a target connection configuration, this connection is normally only visible for your user-id. If you define it as **Shared**, other users can also see the configuration in your registry, provided that they connect to your registry by adding it as a remote registry on their computer.

- Normally, when you disconnect a target connection, the target server (and simulator) are killed because they are no longer needed. In a connection that is flagged as **Shared**, however, they are left running so that other users can connect to them. In other words, you can flag a connection as shared if you want to keep the target server (and simulator) running after you disconnect or exit Workbench.

17. Click **Finish**.

    The target name (in this case, **WRICE_MCF5485**) appears in the **Remote Systems** view.

    The **Disassembly** view opens and a **>BKM>** prompt appears in the **OCD Command Shell**. You have successfully entered Background Mode.

3.3 Configuration Settings

Some Wind River ICE SX configuration options must be changed in order for the Wind River Trace to trace correctly.

For a description of all Trace-specific CF options, see *3.3.1 Configuration Options for Trace*, p.34. For the procedure for setting CF Options, see *3.3.2 Setting Configuration Options*, p.36.

3.3.1 Configuration Options for Trace

AMCC 405, 440, and Freescale ColdFire Processors:

Clear Trace Buffer on GO (TRCCLR)

The **TRCCLR** parameter is used to control where to start saving trace data in the trace memory on a **GO** command. The trace clear settings **YES** and **NO** determine where to start saving the trace data in the trace memory, as explained below.

- **YES** — When a **GO** command is issued, the trace data will be stored in trace memory starting at the first trace memory location. All previously stored trace data will be overwritten and lost. All newly captured trace data will be stored starting at the beginning of the trace memory.

- **NO** — When **GO** command is issued, the trace data will be stored in trace memory starting at the next trace memory location. All previously stored trace data will not be overwritten. All newly captured trace data will be stored starting at the next trace memory location.

Acquire Trace on GO (TRCAQU)

Use the **TRCAQU** parameter to control the acquiring of trace data to the trace memory on a **GO** command. The trace data will also be acquired when stepping, running to a PC value, or running back to a calling function. The trace acquire settings **OFF** and **ON** determine when the trace memory will acquire trace data on a **GO** command as follows:

- **OFF** — When a **GO** command is issued, no trace data will be acquired and saved in the trace memory.
- **ON** — When a **GO** command is issued, all trace data will be acquired and saved in the trace memory.

AMCC 440GX Processor Only:

Trace Output Source (TRCSRC)

Use the **TRCSRC** parameter to determine the source of trace output. Trace can be collected from the General Purpose Input/Output (**GPIO**) or the External Bus Master Interface (**EBMI**.) The default is **GPIO**.

ColdFire Processors Only:

Emit Operands on DDATA Pins (TRCREPORT)

Use the **TRCREPORT** parameter to select whether or not the ColdFire processor will emit read and write memory operands on the DDATA pins.

NO -- When a **GO** command is issued, the CSR register is not configured to force the processor to emit read and write memory operands on the DDATA pins.

YES -- When a **GO** command is issued, the CSR register is configured to force the processor to emit read and write memory operands on the DDATA pins. The memory operands are captured in the trace memory and displayed in the trace as memory cycles. When this parameter is set to **YES**, you must configure the Set Code Range (**CR**) option.

Code Range (CR)

Use the **CR** parameter to set the code range. The code range must be set accurately when the **CF TRCREPORT** parameter is set to **YES**. The code range settings **BASE** and **RANGE** set the base address and the range that the code runs in.

BASE -- Any valid address from 0 to FFFFFFFE that is word aligned. Enter the value as a hex address. (Do not enter a leading **0x** before the address.)

RANGE -- any valid range that, when added to the **BASE** setting, is not greater than FFFFFFFE. Enter the value as a hex address. (Do not enter a leading **0x** before the address.)

Set Work Space (WSPACE)

Use the **WSPACE** parameter to reserve an area of RAM to be used as a workspace. Use the workspace to load a debug exception handler for filtered tracing. You can also use the workspace to load a flash programming algorithm when programming flash memory.

BASE -- Any valid address in RAM that is word aligned. Enter the value as a hex address. (Do not enter a leading **0x** before the address.)

SIZE --Any valid size that, when added to the **BASE** setting, is not greater than FFFFFFFE. Enter the value as a hex address. (Do not enter a leading **0x** before the address.) The debug exception needs a **SIZE** value of 400 in hex.

Set Vector Base Register on IN Command (SET_VBR)

Use the **SET_VBR** parameter to set the Vector Base Register (VBR) to a specified value on an **IN** command. The value should be the same as what the code will set it to. The VBR needs to be set so that the debug exception handler address can be entered in the exception table. The default value is zero. Bits 0..19 are not used.

3.3.2 Setting Configuration Options

There are three ways to set configuration options:

- The **Configure Trace** button in the **Trace** view
- The **CF Options** view
- Low-level commands in the **OCD Command Shell**.

Setting CF Options With the Configure Trace Button

Clicking on the **Configure Trace** button in the **Trace** view opens the **Configure Trace** dialog (see Figure 3-6). This dialog contains fields for all trace-specific CF options for your target architecture.

Trace-specific CF options vary between architectures. The **Configure Trace** dialog shows only those options for your target architecture. For descriptions of all trace-specific CF options, see *3.3.1 Configuration Options for Trace*, p.34.

When you have set all fields in the **Configure Trace** dialog, click **OK** to save your changes.

Setting CF Options With the CF Options View

You can set both trace-specific CF options and all other CF options with the **CF Options** view, by using the following steps:

1. Select **Window > Show View > CF Options**.
2. In the list of available CF options, select **Acquire Trace on GO** (**TRCAQU**). In the **Current Setting** field, select **ON**.
3. In the list of available CF options, select **Clear Trace Buffer on GO** (**TRCCLR**). In the **Current Setting** field, select **YES** or **NO**, as desired.

AMCC 440GX processor only:

a. In the list of available CF options, select **Trace Output Source** (**TRCSRC**). In the **Current Setting** field, select **GPIO** or **EBMI**, as desired.

ColdFire processors only:

a. In the list of available CF options, select **Set Work Space** (**WSPACE**). Select the **Current Setting** field and enter the base address of the workspace and the size of the workspace in bytes. For example, entering **5000 400** will set the workspace to a base address of 0x00005000 and a size of 0x400.

   A 1K workspace in RAM is required to load a debug exception handler for filtered trace with ColdFire. Configure the **Emit Operands on DDATA Pins** option.

b. In the list of available CF options, select **Emit Operands on DDATA Pins** (**TRCREPORT**). In the **Current Setting** field, select **YES** or **NO**, as desired. If you set this option to **YES**, you will also have to configure the **Code Range** option in the next step.

c. Configure the **Code Range** option.

If you set the **Emit Operands on DDATA Pins** option to **NO**, skip this step.

If you set the **Emit Operands on DDATA Pins** option to **YES**:

In the list of available CF options, select **Code Range**. Select the Current Setting field and enter the code range base address and the size of the code range in bytes. For example, entering **400 1000** will set the code range base address to 0x00000400 with a range of 0x1000 bytes.

d. Configure the **Set VBR Register on IN Command** option.

   The exception table base address needs to be configured in RAM so that the debug exception handler address, which is defined in the **Set Work Space** option in Step a, can be entered in the exception table. In the **CF Options** view, select **Set VBR Register on IN Command.** Double-click on the value in the **Current Setting** field (the default is 00000000) and enter the value you want. For example, entering the value 20000000 will set the Vector Base Register to 20000000 whenever an **IN** command is issued.

4. Click **Send All CF Options to Target**.

Setting CF Options With Low-level Commands

To set trace-specific configuration options with low-level commands, use the following steps:

1. Select **Window > Show View > OCD Command Shell**.
2. To enable trace, enter the following command at the >BKM> prompt:

```
>BKM>cf trcaqu on
```

3. Configure the trace buffer.

   By default, Workbench will not clear the trace buffer on a **GO** command. If you want to set Workbench to clear the trace buffer on a **GO** command, enter

```
>BKM>cf trcclr yes
```

AMCC 440GX processor only:

To collect trace from general-purpose input/output, enter

```
>BKM>cf trcsrc gpio
```

To collect trace from the external bus master interface, enter

```
>BKM>cf trcsrc ebmi
```

ColdFire processors only:

a. Configure the **Set Workspace** option.

   A 1K workspace in RAM is required to load a debug exception handler for filtered trace with ColdFire. Set the workspace in the **OCD Command Shell** using the syntax

   `CF WSPACE` *base_addr size*

   base_addr is the base address of the workspace.

   size is the size of the workspace in bytes.

   For example, the command

   ```
   >BKM>CF WSPACE 5000 400
   ```

   will set the workspace to a base address of 0x00005000 and a size of 0x400.

b. Configure the **Emit Operands on DDATA Pins** option.

   Enter

   ```
   >BKM>cf trcreport yes
   ```

   or

   ```
   >BKM>cf trcreport no
   ```

   If you set this option to **YES**, you will also have to configure the **Code Range** option in the next step.

c. Configure the **Code Range** option.

   If you set the **Emit Operands on DDATA Pins** option to **NO**, skip this step.

   If you set the **Emit Operands on DDATA Pins** option to **YES**, you must configure the **Code Range** option using the syntax

   `CF CR` *base_addr range*

   base_addr is the code range base address.

   range is the size of the code range in bytes.

   So entering the command

   ```
   >BKM>CF CR 400 1000
   ```

   will set the code range base address to 0x00000400 with a range of 0x1000 bytes.

d. Configure the **Set VBR Register on IN Command** option.

   Set the value of the VBR register using the syntax

```
cf set_vbr value
```

value is the exception table base address, which needs to be configured in RAM so that the debug exception handler address, which is defined in the **Set Work Space** option in Step a, can be entered in the exception table.

For example, entering the command

```
>BKM>cf set_vbr 20000000
```

will set the Vector Base Register to 20000000 whenever an **IN** command is issued.

4. Reset the target to make your changes take effect:

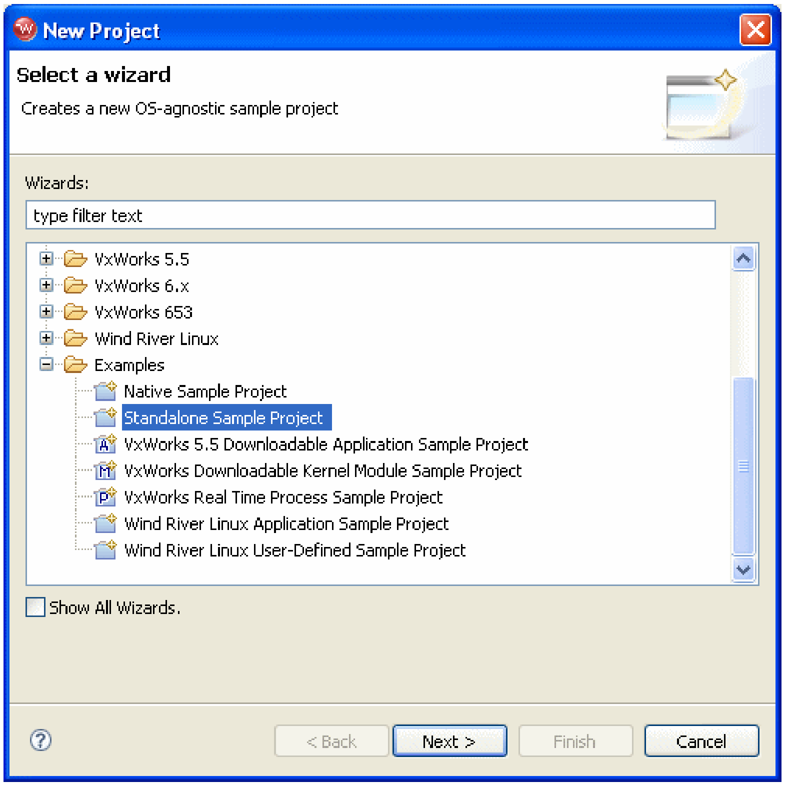
```
>BKM>in
```

3.4 Setting Up a Project

Several example projects are included in Wind River Workbench for demonstration purposes. To open a new demonstration project, use the following steps:

1. In the Workbench toolbar, select **File > New > Project**.

   The **New Project** wizard appears.



Select a wizard

Creates a new OS-agnostic sample project

Wizards:

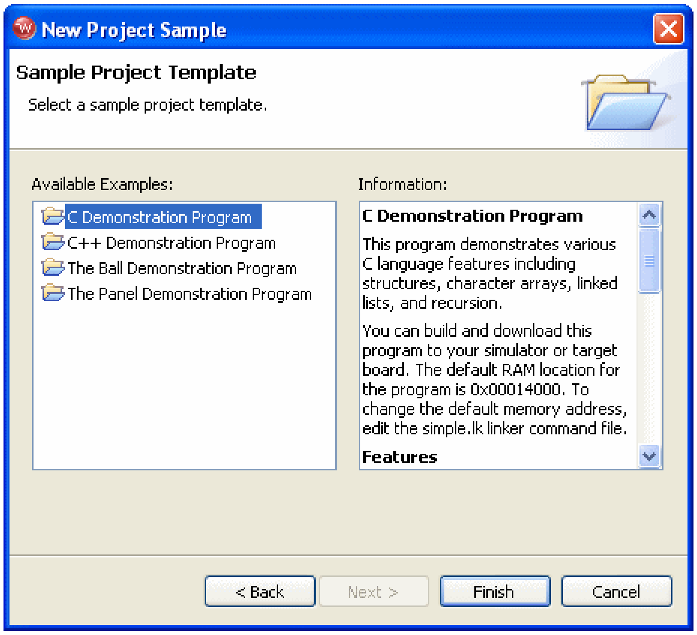
type filter text

- VxWorks 5.5
- VxWorks 6.x
- VxWorks 653
- Wind River Linux
- Examples
  - Native Sample Project
  - Standalone Sample Project
  - VxWorks 5.5 Downloadable Application Sample Project
  - VxWorks Downloadable Kernel Module Sample Project
  - VxWorks Real Time Process Sample Project
  - Wind River Linux Application Sample Project
  - Wind River Linux User-Defined Sample Project

Show All Wizards.

2. Expand the **Examples** folder and select **Standalone Sample Project**.
3. Click **Next**.

   A sample project template appears.


New Project Sample
Sample Project Template
Select a sample project template.
Available Examples:
C Demonstration Program
C++ Demonstration Program
The Ball Demonstration Program
The Panel Demonstration Program
Information:
< Back
Next >
Finish
Cancel

C Demonstration Program

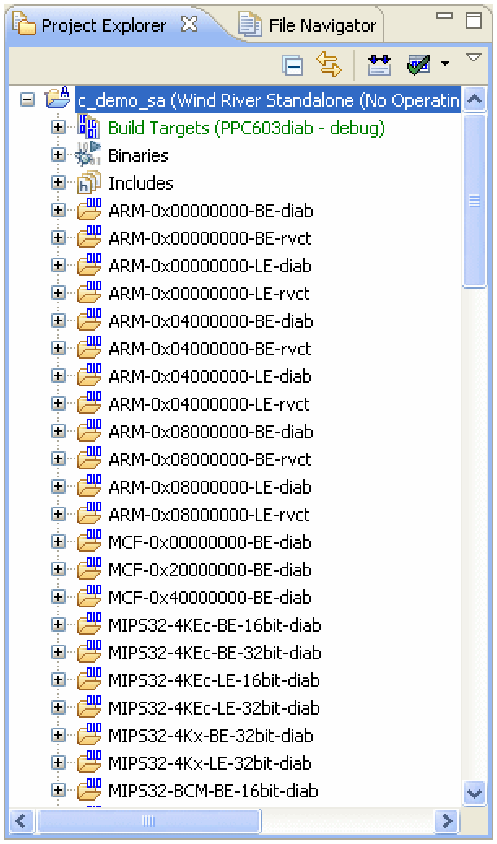
This program demonstrates various C language features including structures, character arrays, linked lists, and recursion.

You can build and download this program to your simulator or target board. The default RAM location for the program is 0x00014000. To change the default memory address, edit the simple.lk linker command file.

Features

4. Select **C Demonstration Program** and click **Finish**.

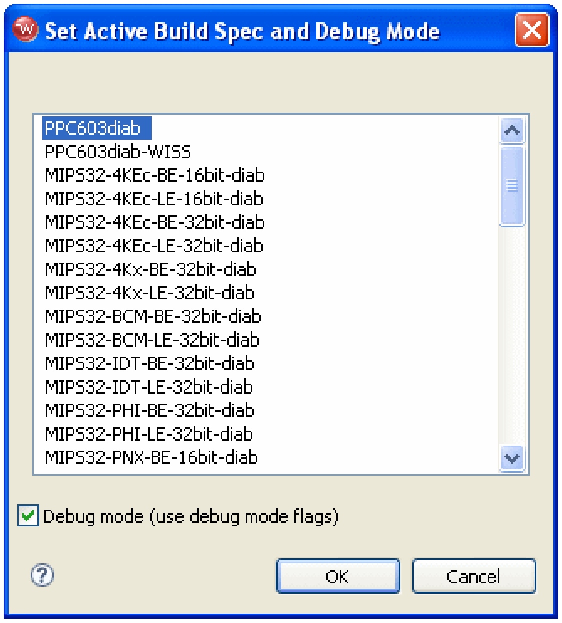
   Workbench creates the sample project in the default **workspace** folder and displays the project **c_demo_sa** in the **Project Explorer** view.



c_demo_sa (Wind River Standalone (No Operatin
Build Targets (PPC603diab - debug)
Binaries
Includes
ARM-0x00000000-BE-diab
ARM-0x00000000-BE-rvct
ARM-0x00000000-LE-diab
ARM-0x00000000-LE-rvct
ARM-0x04000000-BE-diab
ARM-0x04000000-BE-rvct
ARM-0x04000000-LE-diab
ARM-0x04000000-LE-rvct
ARM-0x08000000-BE-diab
ARM-0x08000000-BE-rvct
ARM-0x08000000-LE-diab
ARM-0x08000000-LE-rvct
MCF-0x00000000-BE-diab
MCF-0x20000000-BE-diab
MCF-0x40000000-BE-diab
MIPS32-4KEc-BE-16bit-diab
MIPS32-4KEc-BE-32bit-diab
MIPS32-4KEc-LE-16bit-diab
MIPS32-4KEc-LE-32bit-diab
MIPS32-4Kx-BE-32bit-diab
MIPS32-4Kx-LE-32bit-diab
MIPS32-BCM-BE-16bit-diab

5. To build the sample project for use with your target processor, right-click on the **c_demo_sa** top-level folder and select **Build Options > Set Active Build Spec**.

   The **Set Active Build Spec and Debug Mode** dialog appears.



PPC603diab
PPC603diab-WISS
MIPS32-4KEc-BE-16bit-diab
MIPS32-4KEc-LE-16bit-diab
MIPS32-4KEc-BE-32bit-diab
MIPS32-4KEc-LE-32bit-diab
MIPS32-4Kx-BE-32bit-diab
MIPS32-4Kx-LE-32bit-diab
MIPS32-BCM-BE-32bit-diab
MIPS32-BCM-LE-32bit-diab
MIPS32-IDT-BE-32bit-diab
MIPS32-IDT-LE-32bit-diab
MIPS32-PHI-BE-32bit-diab
MIPS32-PHI-LE-32bit-diab
MIPS32-PNX-BE-16bit-diab

✓ Debug mode (use debug mode flags)

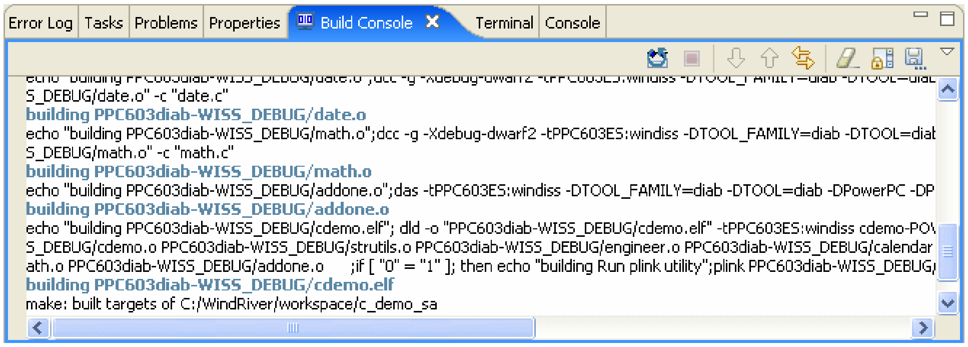
6. Select the build spec for your target processor.

   For AMCC 40x and 44x processors, use the PowerPC build spec (the prefix is **PPC**.) For ColdFire processors, use a ColdFire build spec (the prefix is **MCF**.)

7. Make sure the **Debug mode (use debug mode flags)** checkbox is selected (so Workbench will generate symbolic debug information) and click **OK**.

8. Right-click on the project name and select **Build Project**.

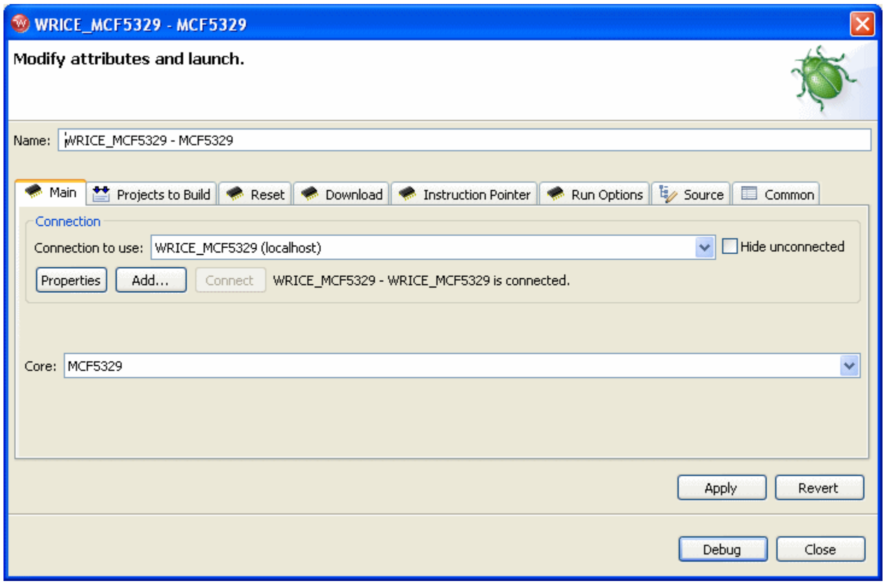
   Workbench builds the sample project. The results of the project build appear in the **Build Console** view.



```
echo "building PPC603diab-WISS_DEBUG/date.o";dcc -g -Xdebug-dwarf2 -tPPC603ES:windiss -DTOOL_FAMILY=diab -DTOOL=diab
S_DEBUG/date.o" -c "date.c"
building PPC603diab-WISS_DEBUG/date.o
echo "building PPC603diab-WISS_DEBUG/math.o";dcc -g -Xdebug-dwarf2 -tPPC603ES:windiss -DTOOL_FAMILY=diab -DTOOL=diab
S_DEBUG/math.o" -c "math.c"
building PPC603diab-WISS_DEBUG/math.o
echo "building PPC603diab-WISS_DEBUG/addone.o";das -tPPC603ES:windiss -DTOOL_FAMILY=diab -DTOOL=diab -DPowerPC -DP
building PPC603diab-WISS_DEBUG/addone.o
echo "building PPC603diab-WISS_DEBUG/cdemo.elf"; dld -o "PPC603diab-WISS_DEBUG/cdemo.elf" -tPPC603ES:windiss cdemo-PO
S_DEBUG/cdemo.o PPC603diab-WISS_DEBUG/strutils.o PPC603diab-WISS_DEBUG/engineer.o PPC603diab-WISS_DEBUG/calendar
ath.o PPC603diab-WISS_DEBUG/addone.o      ;if [ "0" = "1" ]; then echo "building Run plink utility";plink PPC603diab-WISS_DEBUG/
building PPC603diab-WISS_DEBUG/cdemo.elf
make: built targets of C:/WindRiver/workspace/c_demo_sa
```

9. In the Project Explorer, expand **Build Targets**. Right-click on the **cdemo.elf** file for your target and select **Reset and Download**.

   The **Reset and Download** view appears.



Modify attributes and launch.

Name: WRICE_MCF5329 - MCF5329

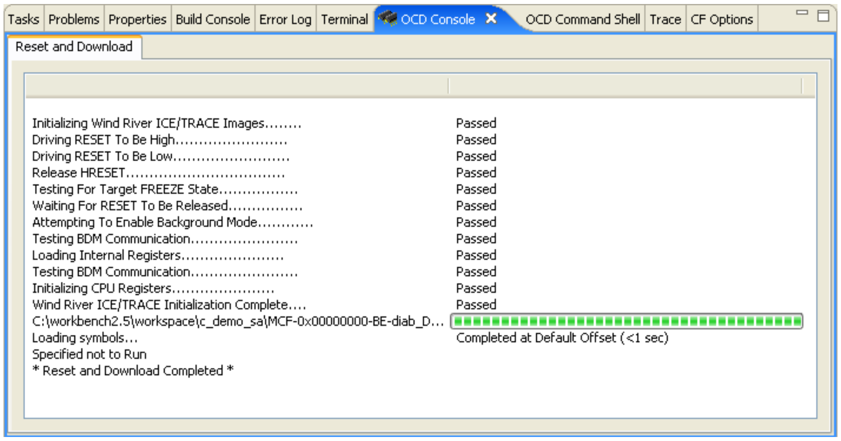
Connection to use: WRICE_MCF5329 (localhost) ☐ Hide unconnected

WRICE_MCF5329 - WRICE_MCF5329 is connected.

Core: MCF5329

10. Leave the settings at their defaults and click **Debug**.

    The **OCD Console** view opens and shows the status of the download operation, as Workbench downloads the sample code to the target.



```
Initializing Wind River ICE/TRACE Images........          Passed
Driving RESET To Be High........................          Passed
Driving RESET To Be Low.........................          Passed
Release HRESET..................................          Passed
Testing For Target FREEZE State.................          Passed
Waiting For RESET To Be Released................          Passed
Attempting To Enable Background Mode............          Passed
Testing BDM Communication.......................          Passed
Loading Internal Registers......................          Passed
Testing BDM Communication.......................          Passed
Initializing CPU Registers......................          Passed
Wind River ICE/TRACE Initialization Complete....          Passed
C:\workbench2.5\workspace\c_demo_sa\MCF-0x00000000-BE-diab_D...
Loading symbols...                                        Completed at Default Offset (<1 sec)
Specified not to Run
* Reset and Download Completed *
```

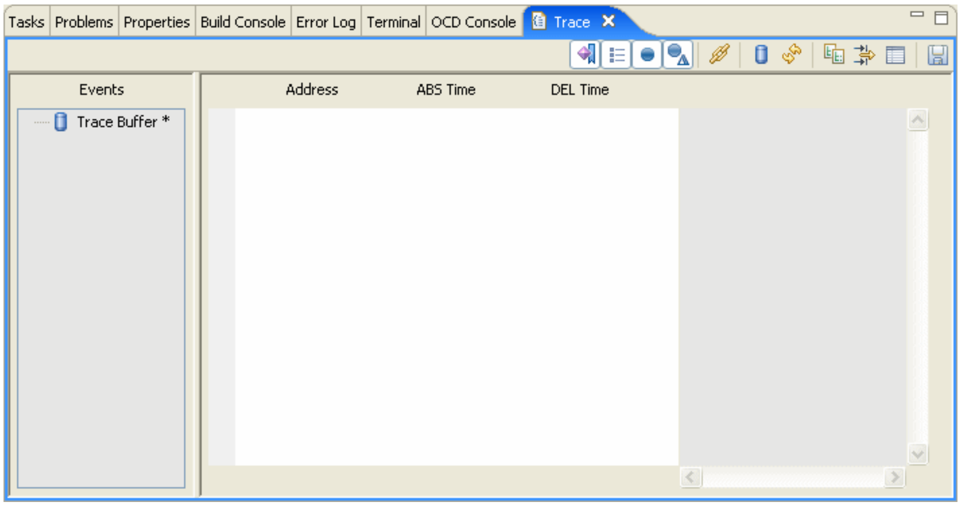
You are now ready to run and trace code.

3.5 Using the Trace View

To open the **Trace** view, select **Window > Show View > Trace View**.

The **Trace** view appears, unpopulated, as shown in Figure 3-1.

Figure 3-1 **Trace View**



The **Trace** view has two fields: the **Events** field and the **Trace** field.

The **Events** field shows the trace buffer. When code runs, the **Events** field shows the start of trace and the end of trace. It also displays the type of trace event.

The Trace field has five columns, from left to right: Event Occurrences (unlabeled), **Address**, **Abs Time, DEL Time**, and Instruction (unlabeled.)

The Event Occurrences column shows the type of trace event.

The **Address** column shows the address or line number of the trace event.

The **Abs Time** column shows the *absolute time*, that is, the elapsed time since the beginning of trace.

The **DEL Time** column shows the *delta time*, that is, the change in absolute time since the last trace entry.

The Instruction column shows the executed instructions. To set the code display, right-click in the Instruction field and select **Show Code Level**. From the list of options, select **Functions**, **Source**, or **Disassembly**.

3.5.1 Trace View Buttons

Table 3-1 shows all the buttons in the **Trace** view toolbar. More detailed descriptions follow the table.

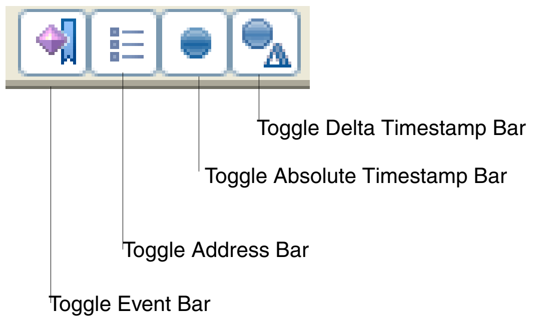
Table 3-1 **Trace View Buttons**

Button	Description
◆	Collapse/expand Event Occurrences column
☰	Collapse/expand **Address** column
●	Collapse/expand **Abs Time** column
⚠	Collapse/expand **DEL Time** column
🔗	Toggle Trace/Source view Auto-Sync
🗑	Clear trace buffer
🔄	Refresh **Trace** view
▤	Open **Trace Rules** dialog
⇨	Filter visible trace events
▦	Configure trace CF options
💾	Save output to file

Collapsing and Expanding Fields

The first four columns in the **Trace** field can be collapsed or expanded using the four toggle buttons in the **Trace** view, as shown in Figure 3-2.

Figure 3-2 **Trace Toggle Buttons**



To collapse any column, click on the toggle button for that field. To re-expand it, click on the toggle button again.

The Instruction column cannot be collapsed.

Toggle Trace/Source view Auto-Sync

Click this button to set the Workbench editor to align itself with any highlighted instruction in the instruction field in the **Trace** view. With this button toggled, clicking on a function in the **Trace** view will cause the editor to jump to that function. To un-sync the **Trace** view and the editor, click the button again.

Clear Trace Buffer

Clear the trace buffer so that previously stored trace data is not included in the next trace that appears. Whenever you add new code, or manually alter the Program Counter value, you should clear the trace buffer before running or stepping, to prevent errors from occurring due to old trace data in the buffer. The button can also be used to trace individual functions by clearing the buffer and then stepping over the function.

The button does not actually flush the trace buffer; it just moves the pointer to the beginning of the buffer, so any previous data is overwritten.

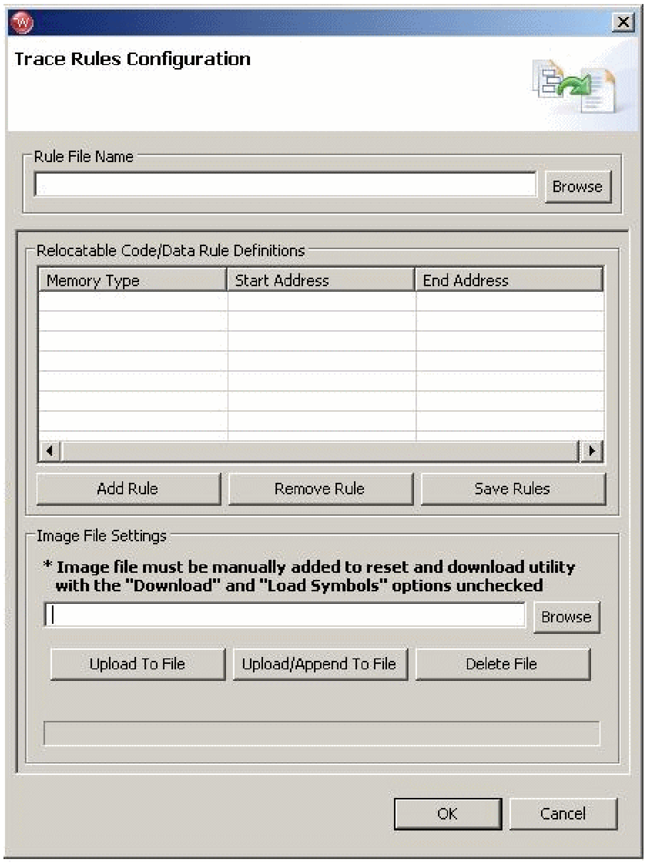
Refresh Trace View

Click this button to refresh the entire **Trace** view, including the **Events** field. Refresh the **Trace** view to display the newest good information.

Open Trace Rules Dialog

Click this button to open the **Trace Rules** dialog, as shown in Figure 3-3.

Figure 3-3 **Trace Rules Dialog**

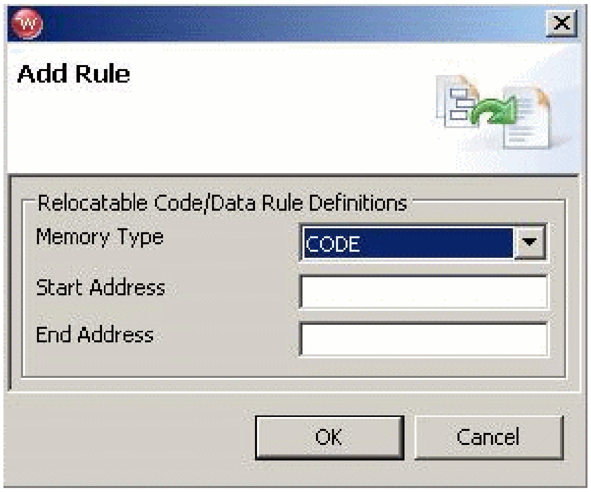


* Image file must be manually added to reset and download utility with the "Download" and "Load Symbols" options unchecked

Use the **Trace Rules** dialog to create a trace rules file, for cases where the code is not running in the address range of the download file specified in the **Reset and Download** view, such as an interrupt service routine in flash.

To create a trace rules file, use the following steps:

1. Click **Add Rule**.



Relocatable Code/Data Rule Definitions
Memory Type: CODE
Start Address
End Address

2. In the **Memory Typ**e field, select **CODE** (for executable code) or **DATA**.
3. In the **Start Address** field, enter the memory address you want trace to begin.
4. In the **End Address** field, enter the memory address you want trace to end.
5. Click **Save Rules**.

   In the browser window that opens, specify a location and name for your rules file. Rules files must be saved as ASCII files with the extension **.rulesconf**.

Next, use the ASCII file you have just created to generate a binary file using the following steps:

1. Click **Upload File**.
2. In the browser window that opens, specify a location and name for your binary file and click **Open**. Workbench will save the binary file with the extension **.elf**.

   If you already have an existing **trace.elf** file, you can click **Upload/Append File** to append the new rule you just created to the end of the existing file.

3. Click **OK** to close the **Trace Rules** dialog.
4. In the **Remote Systems** view, right-click on your target connection name and select **OCD Reset and Download**.
5. The **Reset and Download** view appears.
6. Select the **Files** tab.
7. Click **Add Files**.
8. In the browser window that appears, navigate to the **trace.elf** file you have just created and click **Open**.
9. The **trace.elf** file appears in the file list.
10. Uncheck the **Load Symbols** box. Leave the **Verify** field set to **None**. Make sure the **Download** box is selected.
11. Click **Debug**.

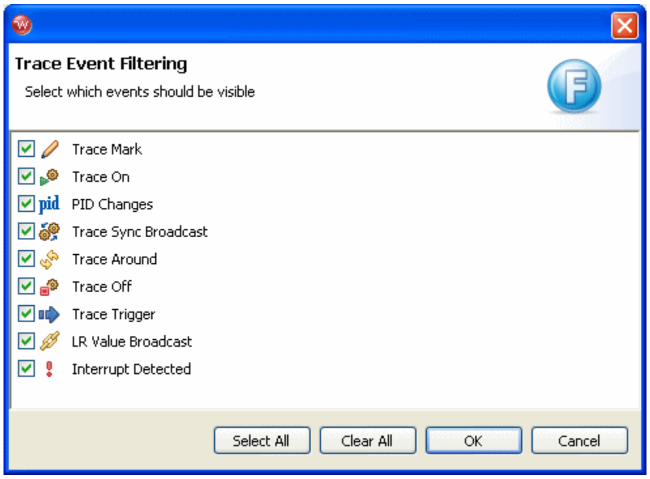
The trace rules are now added to your project.

Filter Visible Trace Events

Click this button to open the **Filter Trace** dialog. Use this dialog to specify which trace events should be visible in the **Events** field in the **Trace** view.

Filtered trace options vary by processor family. For example, Figure 3-4 shows the **Filter Trace** dialog for an AMCC440GP processor, while Figure 3-5 shows the **Filter Trace** dialog for a ColdFire MCF5239 processor.

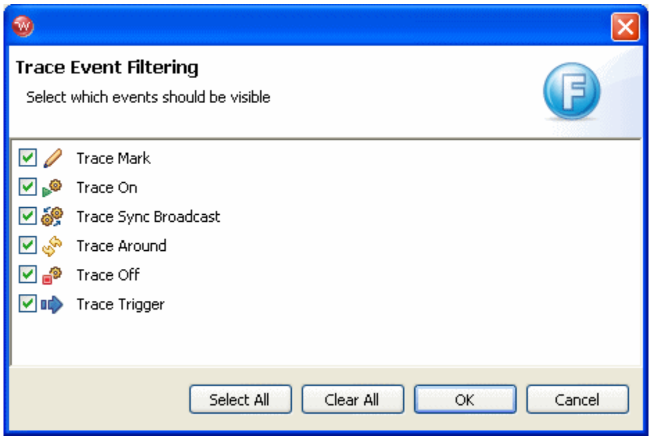
Figure 3-4 **Filtered Trace Options for AMCC440GP**


Trace Event Filtering

Select which events should be visible

- ✓ Trace Mark
- ✓ Trace On
- ✓ PID Changes
- ✓ Trace Sync Broadcast
- ✓ Trace Around
- ✓ Trace Off
- ✓ Trace Trigger
- ✓ LR Value Broadcast
- ✓ Interrupt Detected

Figure 3-5 **Filtered Trace Options for ColdFire MCF5239**


Trace Event Filtering

Select which events should be visible

- ✓ Trace Mark
- ✓ Trace On
- ✓ Trace Sync Broadcast
- ✓ Trace Around
- ✓ Trace Off
- ✓ Trace Trigger

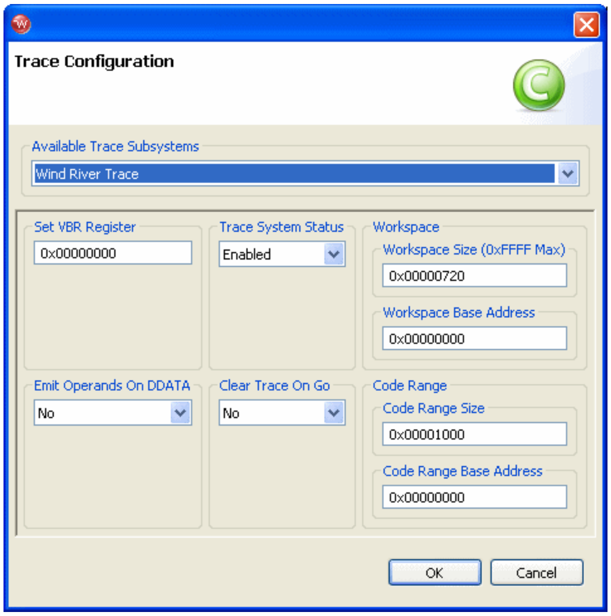
For descriptions of all filtered trace options, see *4. Filtering Trace*.

Configure Trace

Click this button to open the **Configure Trace** dialog. Use this dialog to configure trace-specific configuration options for the Wind River ICE SX.

Configuration options vary by processor family, so the **Configure Trace** dialog will appear differently depending on your target processor. Figure 3-6 shows the **Configure Trace** dialog for ColdFire processors.

Figure 3-6 **Configure Trace Dialog**


Trace Configuration

Available Trace Subsystems: Wind River Trace

Set VBR Register: 0x00000000

Trace System Status: Enabled

Workspace

Workspace Size (0xFFFF Max): 0x00000720

Workspace Base Address: 0x00000000

Emit Operands On DDATA: No

Clear Trace On Go: No

Code Range

Code Range Size: 0x00001000

Code Range Base Address: 0x00000000

For descriptions of all trace-specific configuration options, see *3.3 Configuration Settings*, p.34.

Save Output to File

This button opens a browser window. Use the browser to specify a file to which you can save the information in the **Trace** view.

This button saves the information in the five columns in the **Trace** field (Event Occurrences, Address, Absolute Time, Delta Time, and Instruction). It does not save information from the **Events** field.

3.6 Tracing Execution

Once you have set up your project as described in *3.4 Setting Up a Project*, p.40, you can begin to trace code.

First, decide what trace events you want to be visible in the **Events** field. In the Trace view toolbar, click **Filter Visible Trace Events**.

Available trace events vary by processor family. Examples of available trace events are shown in Figure 3-4 and Figure 3-5. For a description of all trace events, see *4. Filtering Trace*.

By default, all available trace events are selected. Clear the checkbox for any type of trace event you do not wish to be visible.

3.6.1 Setting a Tracepoint

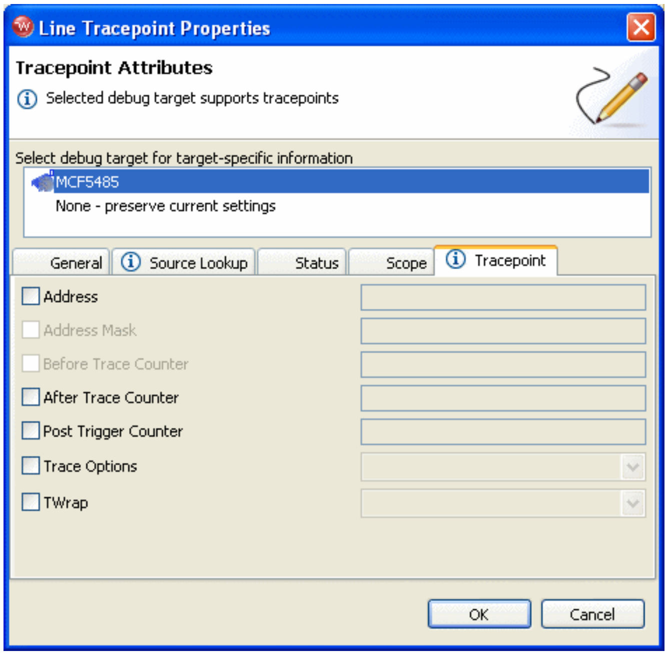
Next, set a tracepoint in your code.

If no tracepoints are set, The trace will contain all code up to the point where the target was suspended, either manually or by hitting a breakpoint.

To set a tracepoint, right-click to the left of the editor (in the gutter) and select **Tracepoints > Add Tracepoint**.

The **Line Tracepoint** dialog appears, as shown in Figure 3-7.

Figure 3-7 **Line Tracepoint Dialog**



Click on the **Tracepoint** tab. Use the **Tracepoint** tab to specify the properties of your tracepoint. The **Tracepoint** tab contains the following options:

Address

Set a tracepoint at a specified address.

Address Mask

Not currently implemented.

Before Trace Counter

Not currently implemented.

After Trace Counter

This counter can be used with either the **Trace Off** or **Trace Around** options to capture more trace data after the matched address.

To use this counter, select **After Trace Counter** and enter a value in the **After Trace Counter** field.

The value you enter is the number of trace records that will be captured after the trace event is detected. The limit is FFFF. The number of instructions per trace record varies by processor family, as follows:

For AMCC 405 processors, instructions per trace record ranges from 1 to 16.

For AMCC 440 processors, instructions per trace record ranges from 1 to 64.

For the V2 and V3 Core ColdFire processors, instructions per trace record ranges from 1 to 8.

For the V4 Core ColdFire processors, instructions per trace record ranges from 1 to 32.

The number of instructions per trace record is normally greater when running with the cache enabled.

You can also set the After Trace Counter using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

```
Get AT
Set AT value
```

value = limit is FFFF.

Post Trigger Counter

This counter can be used only with the **Trace Trigger** option, to capture a specific area of information. If you know exactly where a problem is, a useful combination is to set a **Trigger** (with **Post Trigger Counter**) at that location (code or memory operation).

To use this counter, select **Post Trigger Counter** and enter a value in the **Post Trigger Counter** field. The limit is FFFF.

You can also set the Post Trigger Counter using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

```
Get PT
Set PT value
```

value = limit is FFFF.

NOTE: Under certain conditions the After Trace (**AT**) Counter and the Post Trigger (**PT**) Counter values will be forced to a value, as follows:

Trace Around (TC) on a Data Value Read or Write

If the **AT** is less than 0xF force it to 0xF (ColdFire)

If the **AT** is less than 0x20 force it to 0x20 (AMCC405/440)

Trace Trigger (TG) on a Data Value Read or Write

If the **PT** is less than 0xF force it to 0xF (ColdFire)

If the **PT** is less than 0x10 force it to 0x10 (AMCC405/440)

Trace Off (TD) on a Data Value Read or Write

If the **AT** is less than 0xF force it to 0xF (ColdFire)

If the **AT** is less than 0x20 force it to 0x20 (AMCC405/440)

Trace On (TE) and Trace Off (TD) on an Instruction

If the **AT** is greater than 0x0 force it to 0x0 (ColdFire)

If the **AT** is greater than 0x0 force it to 0x0 (AMCC405/440)

Trace Around (TC) on an Instruction

If the **AT** is less than 0xF force it to 0xF (ColdFire)

If the **AT** is less than 0x20 force it to 0x20 (AMCC405)

If the **AT** is less than 0x100 force it to 0x100 (AMCC440)

Trace Off (TD) on an Instruction

If the **AT** is less than 0xF force it to 0xF (ColdFire)

If the **AT** is less than 0x10 force it to 0x10 (AMCC405/440)

Trace Trigger (TG) on an Instruction

If the **PT** is less than 0xF force it to 0xF (ColdFire)

If the **PT** is less than 0x10 force it to 0x10 (AMCC405/440)

Trace Options

Use this field to specify what kind of tracepoint you are setting. Enable the tracepoint by selecting the **Trace Options** checkbox, and choose the type of tracepoint you want from the drop-down list. The available types of tracepoints vary by processor family. For a description of all trace events, see *4. Filtering Trace*.

TWrap

This option sets the **Stop Trace When Buffer is Full** flag.

This option can be used only with the **Trace On** option to stop trace capturing when the end of the trace buffer is reached. The default is to continue trace at the end of the buffer, so when you stop the target, the last pieces of trace data should match the current instruction.

If the option is set to **ON**, trace will stop at the end of the buffer.

If the option is set to **OFF**, trace will continue after the end of the buffer and overwrite previous trace, beginning at the start of the buffer.

To set this flag, select the **TWrap** checkbox and choose **ON** or **OFF** from the drop-down list.

You can also set this flag using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

```
Get TWRAP
Set TWRAP [ON,OFF]
```

3.6.2 Tracing Execution

Having specified your tracepoint in the **Line Tracepoint** dialog, click **OK**.

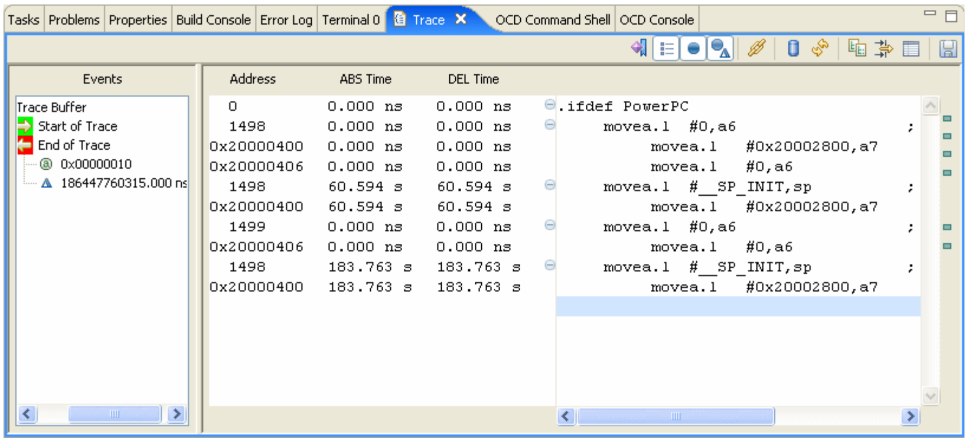
Workbench sets the tracepoint in your code, placing a trace icon in the editor. An entry for the tracepoint appears in the **Breakpoints** view.

In the **Debug** view, click **Resume**. Let some code execute and then click **Suspend**.

In the **Trace** view, click **Refresh View**.

Trace information appears in the **Trace** view, as shown in Figure 3-8.

Figure 3-8 **Trace View**



```
Address     ABS Time    DEL Time
 0          0.000 ns    0.000 ns    .ifdef PowerPC
 1498       0.000 ns    0.000 ns        movea.l  #0,a6                        ;
0x20000400  0.000 ns    0.000 ns                movea.l    #0x20002800,a7
0x20000406  0.000 ns    0.000 ns                movea.l    #0,a6
 1498       60.594 s    60.594 s        movea.l  #__SP_INIT,sp                ;
0x20000400  60.594 s    60.594 s                movea.l    #0x20002800,a7
 1499       0.000 ns    0.000 ns        movea.l  #0,a6                        ;
0x20000406  0.000 ns    0.000 ns                movea.l    #0,a6
 1498       183.763 s   183.763 s       movea.l  #__SP_INIT,sp                ;
0x20000400  183.763 s   183.763 s               movea.l    #0x20002800,a7
```

(In Figure 3-8, the Event Occurrences column is collapsed to make it easier to read the information in the Instruction field.)

4
Filtering Trace

4.1 Overview

The filtered trace capabilities of the Wind River Trace use internal hardware breakpoints to trigger trace capture. You can select an Internal Hardware Code/Data breakpoint and choose the Filtered Trace options to control the trace capture.

NOTE: Filtered trace uses the high speed Personality Module with Auto Voltage Sense. The Auto Voltage Sense features in the Personality Module assume that the customer provides the voltage reference signal.

For the AMCC 405 and 440 processors, this is pin 12 of the 38-pin Mictor connector.

For ColdFire processors, this is pin 9 of the 26-pin BDM connector.

If this signal is missing, you must change the dip switch setting on the Personality Module to match your target's voltage.

To set filtered trace options, click **Filter Visible Trace Events** in the **Trace** view toolbar.

4.2 Filtered Trace Options

Filtered trace options vary by target processor. This section describes the various filtered trace options available.

NOTE: (AMCC 405 and 440 processors only): When any trace event is set using internal hardware breakpoints, the internal and external debug modes are disabled (**DBCR0[IDM**,**EDM**] = **0**). As a result, software breakpoints cannot be used. Any attempt to set a software breakpoint will be ignored. Also, if a normal internal hardware breakpoint is set to halt the target, it will not cause the target to stop.

Interrupt Detected

Display a marker in the **Events** field whenever the processor sends an interrupt.

LR Value Broadcast

Display a marker in the **Events** field whenever the processor broadcasts the value of the Link Register.

PID Changes

Display a marker in the **Events** field whenever the Process ID (PID) of the running process changes.

Report Only

This option will tell the processor to output a special trigger pattern, which will be captured, stored in the trace buffer, and reported as a matched "of" address. This option is only valid with the instruction code breakpoint.

This option can also be set using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

`IHBC` *addr* **`MRK`**

Trace Around

Toggles trace on at the matched address, and immediately toggle trace off again. It will capture only the matched address. This option is useful for measuring the timestamp for a specific routine. Use the After Trace Counter to control the amount of trace captured after the address match.

This option can also be set using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

`IHBC` *addr* **`TC`**
`Set AT` *value*

or

`IHBD` *addr data* **`TC`**
`Set AT` *value*

value = limit is FFFF.

Trace Mark

Place a trace marker on a specified address.

Trace On

Starts capturing trace when the matched address is hit. The trace capture will be disabled until there is an address match.

You can stop trace capturing to preserve the starting matched address by enabling the **Trace Until Buffer Is Full** flag. You can enable this flag either by checking the **TWrap** box in the **Line Tracepoint** dialog and selecting **ON**, or by entering the low-level command **SET TWRAP ON** at the **>BKM>** prompt in the **OCD Command Shell**.

You can also pair the **Trace On** option with the **Trace Off** option to capture a block of trace.

This option can also be set using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

```
IHBC addr TE
```

or

```
IHBD addr data TE
```

Trace Off

This option will stop capturing trace when there is an address match. Use the After Trace Counter to control the amount of trace captured after the address match.

This option can also be set using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

```
IHBC addr TD
SET AT value
```

or

```
IHBD addr data TD
SET AT value
```

value -- limit is FFFF.

Trace Sync Broadcast

This is an internal Wind River option.

Trace Trigger

This option is similar to **Trace Off**. The only difference is that once it hits the trigger and stops tracing, the trace will remain stopped until the next **GO** command is issued. Use the Post Trigger Counter to control the amount of trace captured after the address match.

This option can also be set using low-level commands. At the **>BKM>** prompt in the **OCD Command Shell**, use the following commands:

Syntax:

```
IHBC addr TRIG
Set PT value
```

or

```
IHBD addr data TRIG
Set PT value
```

value -- limit is FFFF.

4.3 Filtered Trace Limitations

4.3.1 Filtered Trace Limitations For Wind River Trace With AMCC 405 and 440 Processors

The filtered trace options that are supported with the AMCC 405 and 440 processors are as follows:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Trace On (**TE**) and one Trace Off (**TD**)
- One To Four Report Only (**MK**)

- One Trace Around (**TC**)
- One To Four Trace Trigger (**TR**)

NOTE: When using both a Trace On (**TE**) and Trace Off (**TD**), the processor must execute the instruction at the Trace On address before the instruction at the Trace Off address. This is because the Wind River Trace module will treat the first trace event it encounters as the Trace On event.

If the code to trace contains a loop, then both the Trace On and Trace Off instruction addresses must be in the loop and the processor must execute the Trace On followed by the Trace Off every time it loops. If the Trace On code executes more than once before the next Trace Off code executes, you will not be able to get a valid trace. Similarly, if the Trace Off code executes more than once before the next Trace On code executes, you will not be able to get a valid trace.

When tracing a small amount of instructions (less than 32) the Trace On and Trace Off may not work. If this does occur, use the Trace Around with an After Trace (**AT**) Counter value set.

NOTE: (AMCC 405 and 440 processors only): When any trace event is set using internal hardware breakpoints, the internal and external debug modes are disabled (**DBCR0**[**IDM**,**EDM**] = **0**). As a result, the software breakpoints cannot be used. Any attempt to set a software breakpoint will be ignored. Also, if a normal internal hardware breakpoint is set to halt the target, it will not cause the target to stop.

NOTE: When performing a filtered trace, do not refresh the **Trace** view while the target is running. If the **Trace** view is refreshed while the target is running, the trace capture will be temporarily halted. After refreshing, the trace capture will resume. During the time the trace capture was halted, the condition that caused the trace to either toggle on or toggle off may have changed, because the target is still running. However, the trace capture will continue from its original state before it was temporarily halted; this may no longer be valid, and may result in an invalid trace being displayed.

4.3.2 Filtered Trace Limitations For Wind River Trace With ColdFire Processors

ColdFire processors do not have a unique trace event PST[0:3] encoding to indicate a trace event has occurred. So in order to generate a trace event, Wind River Trace uses the ColdFire debug exception with a WDDATA instruction to generate a unique trigger event on the PST[0:3]/DDATA[0:3] lines. The debug exception is written to the address defined by the **CF WSPACE** option. The size of the WSPACE must be at least 1K bytes long and should be located in a free area of RAM. Also, the exception vector table entries at the offset addresses **0x30** and **0x31** are written with the address of the debug exception. The Vector Base Register (VBR) value must also be set to a location in RAM prior to running code. The configuration option **CF SET_VBR** should be set to the same value that the code would set it to in RAM, so that the exception vector table entries at offsets **0x31** and **0x32** can be set to the address of the debug exception. The internal hardware code breakpoint registers are used to generate the debug exception. The debug exception is executed every time the event address in the internal hardware breakpoint registers is executed.

As a result of the specific behavior of the V2, V3 and V4 Core ColdFire processors, there are some limitations to filtered trace when using Wind River Trace.

V2 Core

Revision A Debug Modules:

MCF5202, MCF 5204, MCF5206, MCF5206E, MCF5214, MCF5216, MCF5232, MCF5233, MCF5234, MCF5235, MCF5249, MCF5249L, MCF5250, MCF5251, MCF5270, MCF5271, MCF5272, MCF5274, MCF5274L, MCF5275, MCF5275L, MCF5280, MCF5281, and MCF5282

Unfortunately, the V2 core, with a Revision A Debug Module, keeps re-entering the debug exception immediately after executing the RTE instruction at the end of the debug exception. Thus the V2 Core, with a Revision A Debug Module, gets stuck endlessly executing one debug exception after another, making it unusable for Wind River Trace. As a consequence, there are no filtered trace options supported by Wind River Trace for any ColdFire processors with a V2 Core and a Revision A Debug Module.

Revision B+ Debug Modules:

MCF5207, MCF5208A, MCF5211, MCF5212, and MCF5213

MCF52221, MCF52223, MCF52230, MCF52231, MCF52233, MCF52234, and MCF52235

A debug exception can be used for filtered trace with the V2 Core with a Revision B+ Debug Module, since it does not have the debug exception problem of the Revision A Debug Module. The V2 Core, with a Revision B+ Debug Module, has four internal hardware code breakpoints.

The following filtered trace options are supported for ColdFire processors with a V2 Core and a Revision B+ Debug Module:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Trace On (**TE**) and one Trace Off (**TD**)
- One to Four Report Only (**MK**)
- One Trace Around (**TC**)
- One to Four Trace Trigger (**TR**)

V3 Core

Revision B Debug Modules:

MCF 5307, MCF5307A, and MCF5307B

A debug exception can be used for filtered trace with the V3 Core with a Revision B Debug Module, since it does not have the debug exception problem of the V2 Core. There is only one internal hardware code breakpoint in the V3 Core with a Revision B Debug Module. Unfortunately, trying to use a WDEBUG instruction to modify the internal hardware code breakpoint registers inside the debug exception to simulate more than one internal hardware breakpoint does not work. This is because the execution of a WDEBUG instruction within the debug exception causes the BDM debug port to stop communicating with Wind River ICE SX.

The following filtered trace options are supported for ColdFire processors with a V3 Core and a Revision B Debug Module:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Report Only (**MK**)
- One Trace Around (**TC**)
- One Trace Trigger (**TR**)

Revision B+ Debug Modules:

MCF5327, MCF5328, MCF5329, MCF5372, MCF5372L, MCF5373, MCF5373L

A debug exception can be used for filtered trace with the V3 Core with a Revision B+ Debug Module. The V3 Core, with a Revision B+ Debug Module, has four internal hardware code breakpoints.

The following filtered trace options are supported for ColdFire processors with a V3 Core and a Revision B+ Debug Module:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Trace On (**TE**) and one Trace Off (**TD**)
- One to Four Report Only (**MK**)
- One Trace Around (**TC**)
- One to Four Trace Trigger (**TR**)

V4 Core

Revision C Debug Modules:

MCF5407

A debug exception can be used for filtered trace with the V4 Core with a Revision C Debug Module, since it does not have the debug exception problem of the V2 Core. The V4 core has four internal hardware code breakpoints.

The following filtered trace options are supported for ColdFire processors with a V4 Core and a Revision C Debug Module:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Trace On (**TE**) and one Trace Off (**TD**)
- One to Four Report Only (**MK**)
- One Trace Around (**TC**)
- One to Four Trace Trigger (**TR**)

Revision D Debug Modules:

MCF5470, MCF5471, MCF5472, MCF5473, MCF5474, MCF5475, MCF5480, MCF5481, MCF5482, MCF5483, MCF5484, and MCF5485

A debug exception can be used for filtered trace with the V4 Core with a Revision D Debug Module, since it does not have the debug exception problem of the V2 Core. The V4 core has four internal hardware code breakpoints.

The following filtered trace options are supported for ColdFire processors with a V4 Core and a Revision C Debug Module:

- One Trace On (**TE**)
- One Trace Off (**TD**)
- One Trace On (**TE**) and one Trace Off (**TD**)
- One to Four Report Only (**MK**)
- One Trace Around (**TC**)
- One to Four Trace Trigger (**TR**)

NOTE: When using both a Trace On (**TE**) and a Trace Off (**TD**), the processor must execute the instruction at the Trace On address before the instruction at the Trace Off address. This is because the first trace event encountered will be treated as the Trace On event by the Wind River Trace module.

If the code to trace contains a loop, then both the Trace On and Trace Off instruction addresses must be in the loop and the processor must execute the Trace On followed by the Trace Off every time it loops. If the Trace On code executes more than once before the next Trace Off code executes, you will not be able to get a valid trace. Similarly, if the Trace Off code executes more than once before the next Trace On code executes, you will not be able to get a valid trace

When tracing a small amount of instructions (less than 32) the Trace On and Trace Off may not work. If this does occur, use the Trace Around with an After Trace (**AT**) Counter value set.

NOTE: When performing a filtered trace, do not refresh the **Trace** view while the target is running. If the **Trace** view is refreshed while the target is running, the trace capture will be temporarily halted. After refreshing, the trace capture will resume. During the time the trace capture was halted, the condition that caused the trace to either toggle on or toggle off may have changed, because the target is still running. However, the trace capture will continue from its original state before it was temporarily halted; this may no longer be valid, and may result in an invalid trace being displayed.

5
Wind River Trace Performance

5.1 Introduction

The number of trace instructions in Wind River Trace memory depends on the target processor's operational mode. When the caches are enabled, the number of instructions executed in a given time frame is considerably greater than when the caches are disabled. Wind River Trace stores the trace information on every trace clock cycle. Since Wind River Trace's memory is 512 KB long, the trace memory stores a constant time frame of trace information before it begins to overwrite previously stored trace information. This means that when the caches are enabled, a greater number of instructions executed will be stored in Wind River Trace's memory.

AMCC 405 Processor Performance

The AMCC 405 processor's trace clock runs at one half the speed of the core clock. This means that for every trace clock, the trace information emitted by the AMCC 405 is for two core clock cycles. When the caches are enabled, the trace information can show that two instructions were executed for any trace clock cycle. This means that with both the instruction cache and data cache enabled it is possible to see as many as 640,000 instructions stored in the trace memory.

AMCC 440 Processor Performance

The AMCC 440 processor's trace clock runs at one-quarter the speed of the core clock. This means that for every trace clock, the trace information emitted by the

AMCC 440 is for four core clock cycles. When the caches are enabled, the trace information can show that up to 16 instructions were executed for any trace clock cycle. This means that with both the instruction cache and data cache enabled it is possible to see as many as 900,000 instructions stored in the trace memory.

ColdFire Processor Performance

The Version 2 and Version 3 Core ColdFire trace clocks, on pin 24 of the BDM connector, run at the core clock speed. This means that for every trace clock, the trace information emitted by the Version 2 or Version 3 Core ColdFire processors is for one clock cycle. When the caches are enabled, it is possible to see as many as 194,000 instructions stored in the trace memory.

The Version 4 Core ColdFire trace clock, on pin 24 of the BDM connector, runs at one half the core clock speed. This means that for every trace clock, the trace information emitted by the Version 4 Core ColdFire processor is for two clock cycles. When the caches are enabled, it is possible to see as many as 380,000 instructions stored in the trace memory.

A
38-pin Mictor Connector (AMCC 405 and 440 Processors)

A.1 38-Pin Mictor Connector Specifications

The recommended manufacturer's part number for this 38-pin Mictor connector is part number AMP 2-767004-2.

This connector's pin-out information for the AMCC 405 processor is shown in Figure A-1. The pin-out information for the AMCC 440 processor is shown in Figure A-2.

No terminations are required for any of the trace signal pins. For JTAG terminations, the following table contains the Wind River-recommended pull-up/pull-down resistor values that must be placed on the target. Signals not shown should not have pull-ups or pull-downs.

Table A-1 **16-pin Connector JTAG Terminations**

Signal Name	Description
TRST	External 4.7K pull-up

NOTE: These are the termination values for the Wind River reference design. It is important that you verify that the pull-up and pull-down values you use are appropriate for the board that you are working with.

Figure A-1 **38-Pin Mictor Connector Pin-out for the AMCC 405 Processor**

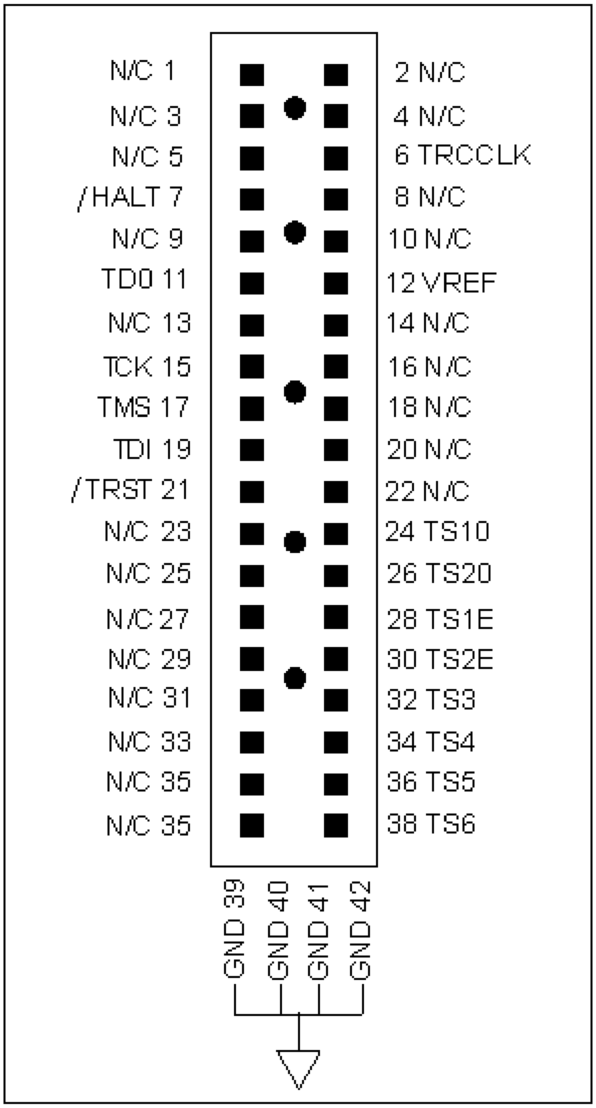
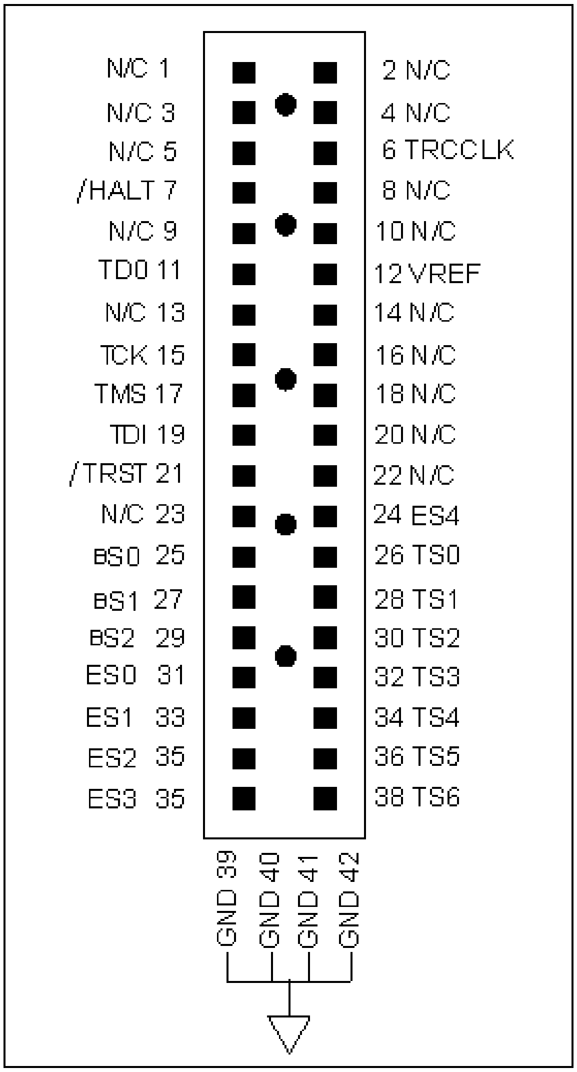


Figure A-2 **38-pin Mictor Connector Pin-out for the AMCC 440 Processor**



A.2 AMCC 405 and 440 Timing Specifications

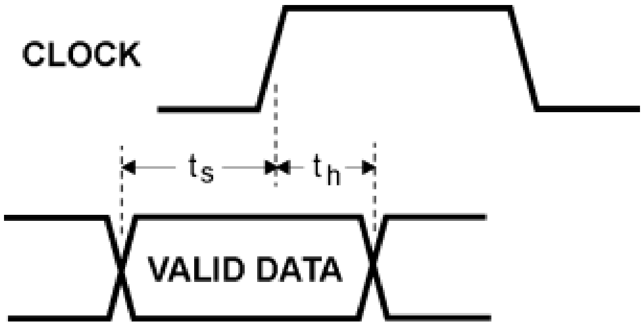


Table A-2 **Key**

Symbol	Parameter	Min	Units
t_s	Setup time	2	nSec
t_h	Hold time	1	nSec

AMCC 405 Processor

38-Pin Mictor Connector

CLOCK	TRCCLK
VALID DATA	TS10, TS20, TS1E, TS2E, TS3, TS4, TS5, andTS6

AMCC 440 processor

38-Pin Mictor Connector

CLOCK: TRCCLK

VALID DATA: TS0, TS1, TS2, TS3, TS4, TS5, TS6, BS0, BS1, BS2, ES0, ES1, ES2, ES3, and ES4

B

26-pin BDM Connector (ColdFire Processors)

B.1 Connector Signal Specifications

The signal pin-out of the ColdFire BDM connector varies by ColdFire processor type. This appendix describes the signal pin-outs for each of the four standard 26 pin ColdFire BDM connectors based on the ColdFire processor type.

Table B-1 breaks down the connector options by processor type.

Table B-1 **ColdFire Connector Options By Processor Type**

Processor	**Connector Option**
MCF5202, MCF5204, MCF5206, MCF5206E	Option One: ColdFire 26-pin BDM Connector
MCF5249, MCF5249L, MCF 5250, MCF 5251, MCF5272, MCF5307, MCF5307A, MCF5307B, MCF5327, MCF5328, MCF5329, MCF5372, MCF5372L, MCF5373, MCF5373L	Option Two: ColdFire 26-pin BDM Connector

Table B-1 **ColdFire Connector Options By Processor Type**

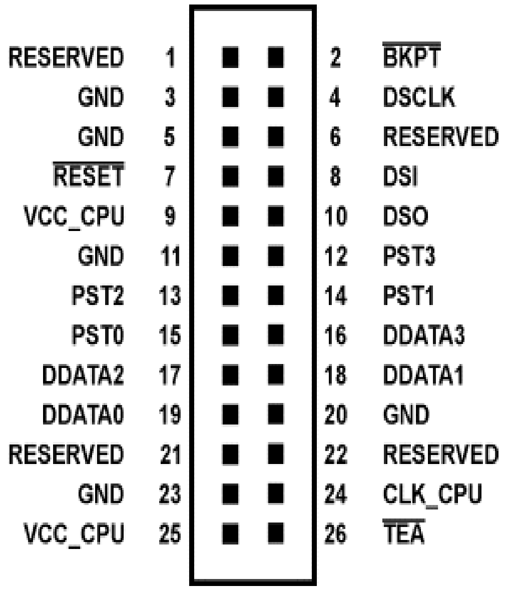
Processor	Connector Option
MCF52221, MCF52223, MCF52230, MCF52231, MCF52233, MCF52234, MCF52235	Option Two: ColdFire 26-pin BDM Connector
MCF5211, MCF5212, MCF5213, MCF5214, MCF5216, MCF5232, MCF5233, MCF5234, MCF5235, MCF5270, MCF5271, MCF5274, MCf5274L, MCF5275, MCF5275L, MCF5280, MCF5281, MCF5282	Option Three: ColdFire 26-pin BDM Connector
MCF5407, MCF5470, MCF5471, MCF5472, MCF5473, MCF5474, MCF5475, MCF5480, MCF5481, MCF5482, MCF5483, MCF5484, MCF5485	Option Four: ColdFire 26-pin BDM Connector

B.1.1 Option One: 26-Pin BDM Connector

- 26 (2 by 13) 0.025" square posts
- 0.10" between centers of adjacent posts
- A sample connector is Samtec part number TSW-113-07-S-D

The pin-outs for the 2 by 13, 0.10" on center ColdFire BDM target connector are as follows:

Figure B-1 **26-Pin BDM Connector: Option One**

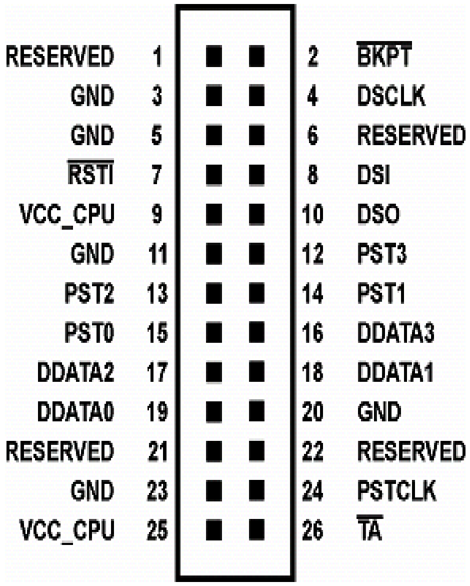


B.1.2 Option Two: 26-Pin BDM Connector

- 26 (2 by 13) 0.025" square posts
- 0.10" between centers of adjacent posts
- A sample connector is Samtec part number TSW-113-07-S-D

The pin-outs for the 2 by 13, 0.10" on center ColdFire BDM target connector are as follows:

Figure B-2 **26-Pin BDM Connector: Option Two**

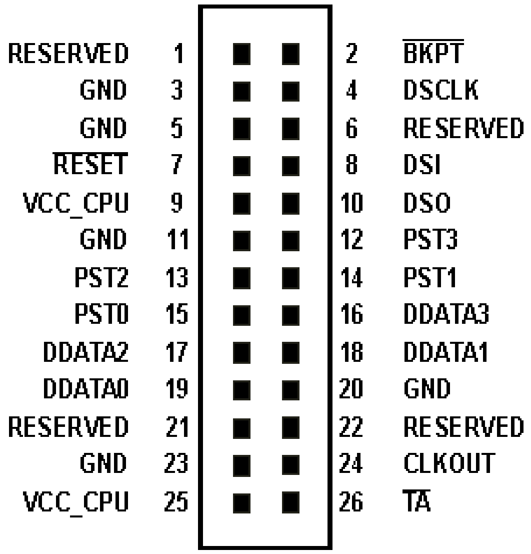


B.1.3 Option Three: 26-Pin BDM Connector

- 26 (2 by 13) 0.025" square posts
- 0.10" between centers of adjacent posts
- A sample connector is Samtec part number TSW-113-07-S-D

The pin-outs for the 2 by 13, 0.10" on center ColdFire BDM target connector are as follows:

Figure B-3 **26-Pin BDM Connector: Option Three**

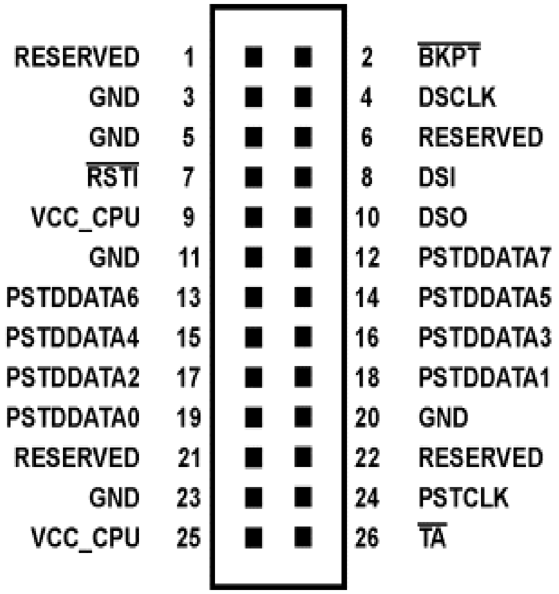


B.1.4 Option Four: 26-Pin BDM Connector

- 26 (2 by 13) 0.025" square posts
- 0.10" between centers of adjacent posts
- A sample connector is Samtec part number TSW-113-07-S-D

The pin-outs for the 2 by 13, 0.10" on center ColdFire BDM target connector are as follows:

Figure B-4 **26-Pin BDM Connector: Option Four**



B.2 ColdFire Timing Specifications

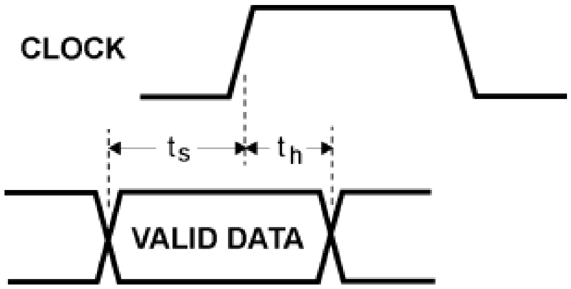


Table B-2 **Key**

Symbol	Parameter	Min	Units
t_s	Setup time	2	nSec
t_h	Hold time	1	nSec

Option One: 26-Pin BDM Connector

CLOCK: CPU_CLK

VALID DATA: PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, and DDATA3

Option Two: 26-Pin BDM Connector

CLOCK: PSTCLK

VALID DATA PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, and DDATA3

Option Three: 26-Pin BDM Connector

CLOCK: CLKOUT

VALID DATA: PST0, PST1, PST2, PST3, DDATA0, DDATA1, DDATA2, and DDATA3

Option Four: 26-Pin BDM Connector

CLOCK: PSTCLK

VALID DATA: PSTDDATA0, PSTDDATA1, PSTDDATA2, PSTDDATA3, PSTDDATA4, PSTDDATA5, PSTDDATA6, and PSTDDATA7

C
Volatility

C.1 Introduction

The Wind River Trace hardware consists of both volatile synchronous static RAM (SSRAM) and an electronically programmable logic device (EPLD).

Table C-1 lists the manufacture and description of the Wind River Trace hardware.

Table C-1 **Wind River Trace Hardware**

Hardware	Manufacturer	Manufacturer's Part Number	Volatile
Programmable logic	Altera	EPF10K130EBC356-iN	Yes
4 MB SSRAM	Cypress	CY7C1339G-133AXC	Yes
(Two devices; 8 MB total)			

The EPLD contains trace signal configuration. No customer proprietary information resides in this memory. All contents are lost when the Wind River Trace is powered down.

All SSRAM contents are lost when the Wind River Trace is powered down.

To power down the Wind River Trace, turn off power to the Wind River ICE SX to which the Wind River Trace is connected.

Index